Cross-Layer Bandwidth Management and Optimization in TDMA Based Wireless Mesh Networks using Network Coding

Vom Fachbereich
Elektrotechnik und Informationstechnik
der Technischen Universität Darmstadt
zur Erlangung des Grades eines
Doktor-Ingenieurs (Dr.-Ing.)
genehmigte

Dissertationsschrift

von

Parag S. Mogre, M. Tech.
Geboren am 11.09.1979, Mumbai, Indien

Erstreferent: Prof. Dr.-Ing. Ralf Steinmetz
Korreferent: Prof. Dr. Albert Banchs
Korreferent: Prof. Dr.-Ing. Matthias Hollick

Tag der Einreichung: 21.05.2010
Tag der Disputation: 08.07.2010

Darmstadt, 2010
Hochschulkennziffer D17

Cross-Layer Bandwidth Management and Optimization in TDMA Based Wireless Mesh Networks using Network Coding

Zur Erlangung des Grades eines Doktor-Ingenieurs (Dr.-Ing.)
genehmigte Dissertation von Parag S. Mogre, M. Tech., geboren am 11.09.1979, Mumbai, Indien
2010 – Darmstadt – D17

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Fachbereich Elektrotechnik
und Informationstechnik

Fachgebiet Multimedia Kommunikation
Prof. Dr.-Ing. Ralf Steinmetz

Cross-Layer Bandwidth Management and Optimization in TDMA Based Wireless Mesh Networks using Network Coding

genehmigte Dissertation von Parag S. Mogre, M. Tech., geboren am 11.09.1979, Mumbai, Indien

Tag der Einreichung: 21.05.2010
Tag der Disputation: 08.07.2010

Erstreferent: Prof. Dr.-Ing. Ralf Steinmetz
Korreferent: Prof. Dr. Albert Banchs
Korreferent: Prof. Dr.-Ing. Matthias Hollick

Technische Universität Darmstadt
Fachbereich Elektrotechnik und Informationstechnik

Fachgebiet Multimedia Kommunikation (KOM)
Prof. Dr.-Ing. Ralf Steinmetz

Zusammenfassung

Drahtlose Mesh-Netze (Wireless Mesh Networks - WMN) stellen eine neue Netzarchitektur dar, um die Abdeckung von Breitbandnetzen zu niedrigen Investitionskosten zu erhöhen. Die Bereitstellung von Dienstgüte (Quality of Service - QoS) ist notwendig, um in WMN anspruchsvolle Multimediaanwendungen zu ermöglichen. Dies muss Ende-zu-Ende im WMN erfolgen und den dort eingesetzten Multihop-Routingverfahren Rechnung tragen. Hierzu ist die Unterstützung von QoS für den Medienzugriff (Medium Access Control - MAC) unabdingbar. Dies manifestiert sich in aktuellen Standardisierungsbemühungen, die Technologien zur QoS-Unterstützung auf MAC-Ebene vorschlagen: Der Mesh-Modus von IEEE 802.16, der Mesh Deterministic Access (MDA) Modus von IEEE 802.11s sowie Wireless HART aus dem Bereich der drahtlosen Sensornetze stellen ausgewählte Beispiele hierfür dar. Diesen Technologien gemeinsam ist die Nutzung von Time Division Multiple Access/Time Division Duplex (TDMA/TDD) um eine explizite Reservierung von Bandbreite zur Datenübertragung für individuelle Verbindungen im WMN zu ermöglichen. Die genannten Technologien ermöglichen die Unterstützung anspruchsvoller Multimediakommunikation und sind damit attraktiv aus Sicht von Netzbetreibern, die durch WMN-Erweiterungen die Netzabdeckung und Leistungsfähigkeit ihrer existierenden kabelgebundenen oder kabellosen Infrastrukturen erhöhen können, ohne dass deutlich höhere Investitionen in Infrastrukturkomponenten erforderlich werden.

Die in drahtlosen Mesh-Netzen verfügbare Bandbreite ist beschränkt. Mit Network Coding wurde jüngst ein Verfahren vorgestellt, das signifikante Bandbreiteneinsparungen verspricht und damit die Netzkapazität von WMN deutlich erhöhen kann. Neben theoretischen Arbeiten zu Network Coding existieren erste praktische Ansätze, um dieses Verfahren in WMN einzusetzen und die real erzielbaren Leistungsgewinne zu untersuchen. Bisherige Arbeiten basieren auf der weit verbreiteten IEEE 802.11-Technologie; entsprechende Untersuchungen in reservierungsbasierten WMN auf Basis von TDMA/TDD existieren bisher nicht. Angesichts des hohen Potentials reservierungsbasierter WMN, die erst Garantien für anspruchsvolle Multimediaanwendungen ermöglichen, erscheint es vielversprechend, Network Coding in der genannten Netzklasse eingehend zu untersuchen.

In dieser Arbeit wird gezeigt, dass existierende, paketbasierte Verfahren des Network Codings in reservierungsbasierten WMN auf Basis von TDMA extrem ineffizient sind. Es werden Designkriterien für Network Coding in entsprechenden WMN hergeleitet. Basierend auf diesen Kriterien wird ein neues Paradigma für Network Coding, das sogenannte „Datenstrom-basierte Network Coding" (*Stream-oriented Network Coding* - SONC), vorgeschlagen.

SONC trifft Codierungsentscheidungen auf der Basis von Datenströmen (*streams*). Hierbei wird ein Datenstrom als Folge von Datenpaketen an einem Netzknoten, der als Relay agiert, definiert. Diese Datenpakete besitzen jeweils gemeinsame Vorgänger- und Nachfolgerknoten. Die Operation auf Datenströmen lässt einen günstigeren Reservierungsaufwand entstehen, da immer ein Strom von Paketen codiert wird. In der Arbeit wird analysiert, wie mögliche und vielversprechende Codierungskonstellationen erkannt werden können und es werden Mechanismen vorgestellt, mit denen diese umgesetzt werden können. SONC nutzt hierzu verteilte Mechanismen und einen schichtenübergreifenden Ansatz. Dies ermöglicht, dass SONC im WMN umgesetzt werden kann, wenn die einzelnen Knoten ausschließlich lokale Statusinformationen zur Verfügung haben. Unter Hinzunahme WMN-globaler Statusinformationen kann die Effektivität von SONC weiter erhöht werden.

In der zweiten Hälfte der Arbeit wird eine entsprechende zentral optimierte Routing-Erweiterung (*Centrally Optimized Routing Extension* - CORE) vorgestellt. Diese erweitert bestehende Routingverfahren in WMN, um gezielt Codierungsmöglichkeiten für SONC herbeizuführen. CORE arbeitet schichtenübergreifend in enger Abstimmung mit SONC. Hierbei erlaubt CORE dem Netzbetreiber, den maximalen Berechnungsaufwand vorzugeben und ermöglicht damit den Betrieb nahezu in Echtzeit. Dies ermöglicht den Einsatz von CORE in realistischen Netzen mit dynamischen Verkehrsanforderungen.

Um die Machbarkeit der entwickelten Lösungsansätze zu zeigen, wurden alle beschriebenen Verfahren im Kontext des Mesh-Modus des IEEE 802.16-Standard prototypisch umgesetzt. IEEE 802.16 wurde stellvertretend für die Klasse der TDMA- und reservierungsbasierten WMN gewählt, die entwickelten Lösungen sind übertragbar auf gleichartige Technologien. Die Ergebnisse der umfangreichen experimentellen Evaluation zeigen, dass signifikante Bandbreitenersparnisse realisiert werden können und somit die Netzkapazität erhöht werden kann, wenn CORE und SONC zum Einsatz kommen.

Abstract

Wireless Mesh Networks (WMNs) provide a novel network architecture to extend broadband network coverage with low costs. Additionally, we see an increased interest in supporting demanding multimedia applications in next-generation wireless mesh networks. Provision of Quality of Service (QoS) in wireless mesh networks requires end-to-end support for routing packets via a suitable multihop route to the destination. However, in wireless mesh networks, if the medium access control layer does not support mechanisms for QoS support at a per-link level, all efforts for providing end-to-end QoS are futile. Hence, we see a trend towards standards for wireless mesh networks which support QoS at the Medium Access Control (MAC) level on a per-link basis. The IEEE 802.16 standard's mesh mode of operation, the IEEE 802.11s Mesh Deterministic Access (MDA) mode of operation, and upcoming sensor network standards such as the Wireless HART standard, support MAC layer QoS mechanisms. A common feature of these standards is the use of Time Division Multiple Access/Time Division Duplex (TDMA/TDD) for supporting QoS, by enabling the explicit reservation of bandwidth for data transmissions on individual links in the wireless mesh network. This has enabled the setup of wireless mesh networks which are able to support hard QoS guarantees, and are thus viable for supporting the highly demanding multimedia traffic which can be expected in such networks in future. This, makes wireless mesh networks using such standards attractive for network operators who want to extend the reach of their current wired networks, as well as cellular wireless networks to support additional traffic, and at the same time not incur exorbitant additional costs for the infrastructure setup.

However, the bandwidth in such wireless mesh networks still remains a scarce resource. Recently, network coding has been investigated as a novel mechanism to permit the saving of valuable bandwidth in such wireless mesh networks for individual transmissions, thereby increasing the traffic carrying capacity of the wireless mesh networks significantly. Beginning from mainly theoretical work, recently we have also seen an effort to investigate the practical gains which can be obtained via deployment of network coding in wireless mesh networks. However, to-date, the practical investigations for deployment of network coding have been limited to wireless mesh networks based on the IEEE 802.11 standard. There have been no significant investigations on the deployment of network coding, and its benefits, in TDMA/TDD based multihop wireless mesh networks. Given, however, the fact that the next generation of wireless mesh networks would be using bandwidth reservation schemes to support advanced multimedia services, it is vital that network coding be investigated in the light of such wireless mesh networks. This work bridges the above gap.

In this thesis we first demonstrate that contemporary packet-by-packet approaches to network coding are highly inefficient in reservation based TDMA WMNs. We derive thus design principles for network coding solutions in such WMNs. Based on our design principles we then go on to propose a new paradigm for network coding, *Stream Oriented Network Coding* (SONC). SONC makes network coding decisions and performs network coding operations at the granularity of *streams*. The term stream is defined in the context of SONC in this thesis to refer to a aggregation of packets arriving at a relaying node from a given prior hop and going on to the same next hop. This enables SONC to amortize the reservation overhead via network coding operations on a range of packets. We present and analyze means to identify suitable opportunities for SONC and also mechanisms to deploy these. SONC uses distributed mechanisms and a cross-layer approach to achieve its goal. Thus nodes in the WMN can deploy SONC with just local

knowledge. However, the effective benefits which can be obtained via deployment of SONC can be further improved with global (WMN-wide) knowledge.

Hence in the second half of the thesis we present *Centrally Optimized Routing Extensions* (CORE) which, as the name suggests, extends the functionality provided by the default routing algorithms in the WMN to enable more opportunities to benefit from SONC. CORE too uses a cross-layer approach and works in close collaboration with SONC. CORE's design is such that the network operator can limit the computational effort to a predefined maximum value, thereby, ensuring that it can be run in near real-time. This enables our CORE to operate in realistic networks with changing traffic demands.

To demonstrate the proof-of-concept of our solutions we have implemented these using the IEEE 802.16 standard's mesh mode of operation as a prototype for reservation based TDMA WMNs. Our results show that the WMN can benefit significantly in terms of bandwidth savings and additional traffic which it may support when CORE and SONC are deployed in the network.

Acknowledgements

I would like to take this opportunity to express my gratitude to all those who made it possible for me to finish this document. Firstly I would like to express my deepest gratitude to my advisor Prof. Ralf Steinmetz for his continuous support, advice and strong encouragement throughout the time I have been involved with the Multimedia Communications Lab led by him. I would like to thank Prof. Albert Banchs and Matthias for their willingness to help with the evaluation of this thesis. At the same time i also wish to thank the entire examination panel (Prof. Anja Klein, Prof. Ralf Steinmetz, Prof. Matthias Hollick, Prof. Albert Banchs, Prof. Hans Eveking, and Prof. Gerd Balzer) for the excellent discussion and insightful comments about the work in the thesis. The Multimedia Communications Lab (KOM) group has been a valuable work environment for me enabling me to continuously develop myself over the years that I have been here. This is also due to the wonderful colleagues here at KOM from whom one can learn a lot. Special thanks are due to Matthias Hollick for his continuous help and support without which I would certainly not have reached the point of writing this acknowledgement. Special thanks are also due to all my team members, Johannes Schmitt, Matthias Kropff, Andreas Reinhardt, and Farid Zaid for their wonderful support and assistance with my work during the final phases of my thesis writeup which made it possible for me to make any progress on the writeup of the thesis. All of them have wholeheartedly made sacrifices in order to help me make progress with the writeup. No words of thanks would be sufficient for this help, still I would like to use this opportunity to thank all my team members. I would like to thank all the other past and present KOM colleagues who have in some or other way left their own imprint on this work. I must express my thanks for the invaluable support given to me by the administrative staff and other colleagues here at KOM, especially (listed using a randomized algorithm): Karola, Moni, Sabine, Frank, Frau Kolb, Frau Ehlhardt, Frau Scholz-Schmidt, Tomas, Jan, Nico, Doreen, Amr, Michael, Nicolas, Lasse, Marek, Birgit, Kalman, Sonja, Manuel, Hannes, Ivan, Vasilios, Aleksandra, Oliver, Ralf, Jens, Irina, Susanne, Julia, Abdu, André[2], Tronje, and Markus. It was thanks to the support of these listed and unlisted colleagues that my experience at KOM has been an enjoyable one. Special thanks are also due to André Miede for his continuous support and help with the template for this thesis. Without his help the thesis would not have appeared in the good layout it appears in today.

I would like to express my gratitude for the support provided to me by the DFG within the framework of the GK-e-Commerce (Prof. Alejandro Buchmann), and Siemens Corporate Technology, IC 2 (Dr. Rainer Sauerwein, Dr. Christian Schwingenschloegl, Andreas Ziller) for funding my research and being very supportive and encouraging with the research work and always providing new impetus to my research.

I thank all my students for giving me the opportunity guide them and also for helping me with my research. Thanks for the assistance (list generated using a randomized algorithm): Boris, Andreas, Christian, Kalman, Vesselina, Damir, Viraj, Gregor, Martin[2], Matthias, Anton, Marcos, Nico, Guillaume, Eduardo, Jesus, Ahmad, Benjamin, Oliver, Zheng, Stefan, Johannes, Nico, Hans Werner, Daniel, Dao, Zdravko, Florian, Frank, Tobias, Mostafa, and Maxim.

In addition I would like to thank all my friends here in Darmstadt for making the stay in Germany an interesting one. Thanks for all the help support and enjoyable times (again listed using a randomized algorithm): Sarla, Inãki, Poonam, Himanshu, Ravi, Rajeev, Debashish, Pani, Bhaskar, Filipa, Divya, Shankar, Ruben, Swati, Kripa, Jan, Anita, Pallavi, Andrea, Rajani, José, Alex, Takeshi. I would like to express my special thanks to Frau Kleinschmidt for all her sup-

port and continuous help (also with learning the German language) without which the stay in Germany would have been much more difficult, if not impossible.

Last but not the least I would like to thank my family for their complete support to my endeavour, without their help and assistance I would certainly not have been able to complete this document.

Contents

Part I

Introduction, Background and Related Work

1 Introduction and Motivation

Network connectivity is rapidly becoming a very critical and vital part of our daily lives. Cheaper computing power, smaller and portable smart devices are enabling people to perform tasks which, in the past, were only possible using relatively bulky fixed computers at home or in the office. This also means that people increasingly communicate for both business and personal reasons while on the move (see Ref. [40] for a detailed urban mobility model). Further, the communication is moving from mainly voice communication to rich multimedia communication. People also want to access the network on the move for leisure and entertainment, e.g. watching television or videos while travelling. A critical factor which is necessary to drive this trend further, and for realizing the full potential of communication enabled services and entertainment, is ubiquitous broadband network coverage. To cater to this demand we see a range of networking standards and technologies which address the above need at different scales.

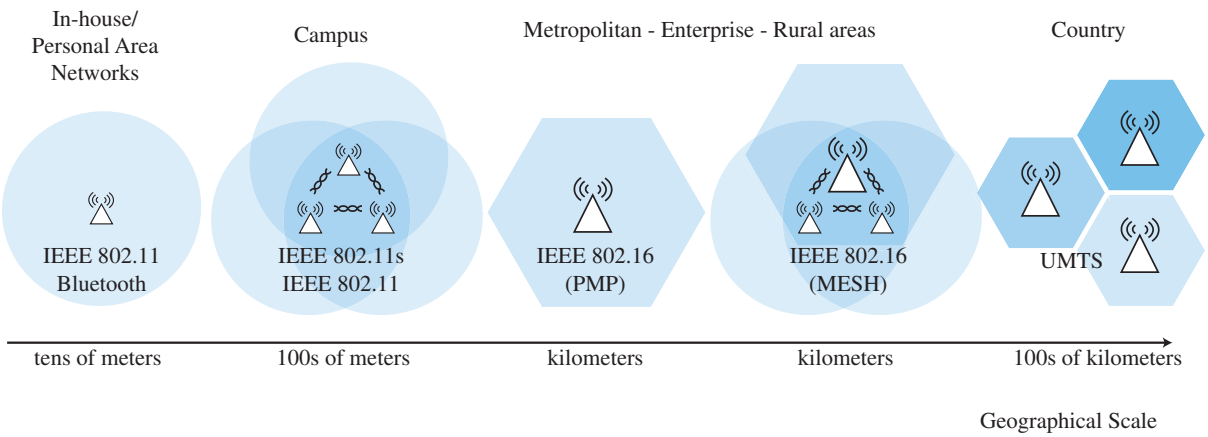


Figure 1.1: Various Network Scales

For example Fig. 1.1 shows the typical network scales observed today and the typical network technologies and standards which have evolved to address networking needs at those respective scales. Note that we are restricting the discussion to wireless networks only and the technologies used to wirelessly access networks which are probably global at scale (e.g. accessing and communicating with systems on the Internet using wireless access via UMTS). We see an ubiquitous use of bluetooth in networking at the level of personal area or body area networks, e.g. communications between a digital personal organizer and a personal computer or laptop. The laptop device may in turn connect to the local area networks via IEEE 802.11 [41] based technologies. However, the latter networks are restricted in the coverage area they offer and the QoS support provided to the applications using these networks. On the other hand we see networks such as UMTS/GSM networks which mainly cater to voice traffic today and are optimized for voice data. These networks offer connectivity to the users on the move and typically at relatively high mobility (speeds up to several hundred kilometers per hour). However, we currently see a gap in the connectivity spectrum. Namely one at the community level, or the level of an enterprise or a metropolitan area. The IEEE 802.16 standard aims to target this gap and specifies different modes of operation. An interesting networking paradigm which has seen a lot of interest in the research community over the past few years is that of Wireless Mesh Networks (WMNs) (see [4] for a detailed survey of WMNs). WMNs are generally considered to be a more flexible and

cheaper alternative to address the coverage gap. The core idea of WMNs is that multiple nodes collaborate to form a wireless backbone which can then be used to route packets from source nodes to the destination nodes possibly via multiple hops. It also enables users to connect to a node on the backbone of the WMN, also using multiple technologies, and then access the resources presented by the WMN backbone to communicate with other nodes on the WMN as well as users connected to nodes on the WMN. In addition, if any node in the WMN backbone happens to have access to the Internet via wired or wireless links all the nodes connected to the WMN and on the WMN backbone can communicate with systems on the Internet. What makes WMNs especially interesting (also from a network provider's perspective) is that these are self organizing and can be setup much cheaper than networks with a wired backbone. If additional coverage is needed it is easy to deploy additional nodes in the needed areas which join the rest of the WMN backbone, making extension of the wireless backbone very flexible. It is additionally possible to involve end user systems in the WMN backbone which leads to further cost benefits for the operator of the network. Given these benefits the IEEE 802.16 standard [42] introduces in addition to the Point to Multipoint (PMP) mode of operation an additional and optional MeSH* mode of operation. When operating in this mode nodes in the network can communicate with each other and the base station via routing packets over multiple hops if needed and the nodes (Subscriber Stations (SS)) do not need to be in the direct range of the Mesh Base Station (MBS). In contrast to the relay mode of operation currently being standardized by the IEEE 802.16 working group, the MeSH mode supports topologies which need not have to be a strictly tree topology rooted at the base station. WMNs are also being followed by interest within the IEEE 802.11 working group which has led to the working group 802.11s which is concerned with extensions to the IEEE 802.11 standard to enable efficient setup of WMNs using the 802.11 standard. The goal here is to be able to connect nearby mobile devices (laptops etc.) to form a local mesh network which may or may not provide access to the Internet. WMNs are also seen as a promising approach to setup flexible industrial sensor networks for factory/process automation. The WirelessHART standard [34] supports the setup of wireless mesh topologies for the latter application scenarios.

Thus, WMNs are a promising technological paradigm to address the future needs for seamless and cheap connectivity. On the other hand we also see a trend towards more demanding applications, in terms of bandwidth needed, delay tolerance, and in general the Quality of Service (QoS) requirements. Multimedia communication is becoming more relevant esp. with cheaper devices which enable efficient capturing of high quality multimedia content, which then needs to transported between the communication partners. Another typical scenario where WMNs need to support stringent QoS requirements is in the area of industrial control networks, and automation. Keeping this in mind the various standardization groups are putting in a lot of effort to specify sophisticated mechanisms to support QoS at the Medium Access Control (MAC) layer. The aim is to provide predictable network performance to the very demanding multimedia traffic expected to form a bulk of the traffic carried by such networks in future. In this regard the IEEE 802.16 standard specifies a rich set of options to support QoS. Its reservation based nature means that unlike the existing 802.11 based WMNs, in WMNs and networks using the IEEE 802.16 standard it is possible to enforce and guarantee very stringent QoS requirements. There are significant efforts to provide throughput guarantees in IEEE 802.11 based WMNs too (see Ref. [9]) and also optimize the performance of such networks (e.g. Ref. [91]).

Although the IEEE 802.16 standard specifies the basic messages to set up QoS support, concrete mechanisms and algorithms to support QoS are left open to allow vendors to optimize and differentiate their products. Similar to the IEEE 802.16 standard the proposed IEEE 802.11s standard which aims to offer extended support for IEEE 802.11 based WMNs also specifies a

* Throughout this document we use the notation "MeSH" when referring to the IEEE 802.16 standard's [42] mesh mode of operation

Mesh Deterministic Access (MDA) (see [36]) mode of operation which also uses bandwidth reservations to support QoS. The WirelessHART specification also uses bandwidth reservation for QoS support. Common to these next-generation standards for supporting WMNs and QoS is the use of Time Division Multiple Access/Time Division Duplex (TDMA/TDD) for the bandwidth reservation. Thus, irrespective of the application domain for WMNs we see a clear trend towards specification of mechanisms for supporting QoS at the MAC layer. This is very vital, as in a wireless mesh network supporting end-to-end QoS is very difficult if not impossible when no QoS is supported at the MAC layer.

However, supporting QoS in WMNs is not the only aim of the network providers. Network providers deploying WMNs to support their future broadband wireless network infrastructure also aim to maximize the traffic carrying capacity of their network. This usually means more revenue for the network provider and hence a faster return on investment. Bandwidth in WMNs is a very critical and valuable resource. In their seminal work in Ref. [32] the authors analytically modelled the capacity and achievable throughput in wireless networks. The findings in [32] showed that under the given assumptions, the throughput obtainable for each node in the wireless network for a randomly chosen destination is $\Theta(W/\sqrt{n \log n})$ bits per second under a noninterference protocol. Where W is the number of bits per second which may be at most transmitted by a node on the wireless medium due to the radio and modulation limitations, and n is the number of nodes in the wireless networks. Thus, one sees that although it is very much essential that more nodes participate in the wireless network from a network provider's point of view, the addition of each node in the multihop wireless network is also associated with a penalty in terms of achievable transport capacity per node in the wireless network. This is because every node in the network needs to share the wireless resources with each other node in its extended neighbourhood. Recently Network Coding (NC) has been introduced as a means to enhance the traffic carrying capacity in networks. Network coding was introduced by the seminal work of Ahlswede et al. in [2]. Their work showed that it is not always beneficial to regard information to be multicast as a fluid which is simply routed or replicated. Rather, they showed that considering the information contained in the data, and using network coding at the nodes in the network, bandwidth may be saved (a more detailed discussion of network coding can be found in Chap. 2). Later work also demonstrated the benefits of using network coding in wireless networks for the case of multiple unicast flows, i.e. flows between source destination pairs in the wireless mesh network (e.g. [49, 48, 52]). Findings show that network coding is able to provide throughput gains even in the case of multiple unicast sessions in the wireless mesh network. Thus, network coding is a very promising tool for network providers to increase the capacity of their wireless mesh networks.

However, most of the work in the area of network coding has either been only of theoretical nature, or has been designed and developed keeping the legacy IEEE 802.11 based MAC layer in mind. Network coding has not been thoroughly studied in the scope of WMNs using reservation based MAC layers, like e.g. the IEEE 802.16 MeSH mode. In this work we bridge the above chasm, by designing and developing efficient solutions for practically deploying network coding within the scope of wireless mesh networks using reservation based MAC layers. To present the research in concrete settings and to present a proof-of-concept for the developed results we use in this document the MeSH mode as a prototype for next-generation MAC layers using bandwidth reservation for supporting QoS at the MAC layer. However, the developed results are generic and hold for other similar wireless mesh networks too. In particular, we look at network coding in reservation based WMNs for enhancing the traffic carrying capacity of the network. Sec. 1.1 provides details about the scope of the investigations in this document. Then in Sec. 1.2 we list the contributions of this work also identifying the approach used to solve the problem to be tackled. Finally, in Sec. 1.3 we provide an outline of the remainder of this

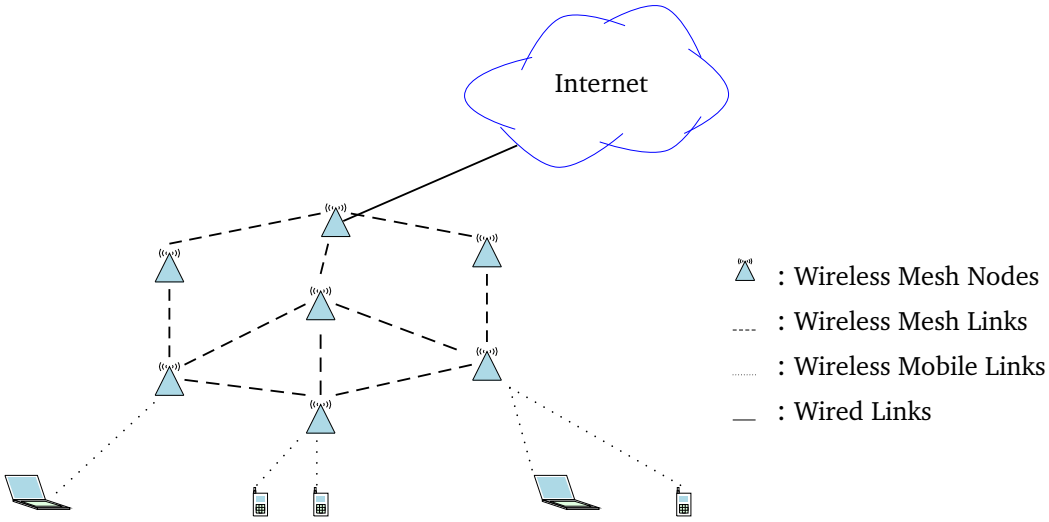


Figure 1.2: Wireless Mesh Network Deployment Scenario

document which will enable the readers to follow the discussion and orient themselves when reading the document.

1.1 Problem Domain

In this section we outline the scope of the research, identify the concrete setting for the research. Here, we also outline the assumptions made, and give an outline of the concrete research goals being addressed in this thesis.

In this thesis we focus on reservation based WMNs, where bandwidth reservations at the MAC layer are used to guarantee QoS requirements of applications. Such networks can provide a wireless backbone to which mobile users can connect using Local Area Network (LAN) and Wireless Local Area Network (WLAN) technologies. The wireless mesh network can thus provide a wireless backbone which allows the mobile nodes to access the Internet. This is achieved by possibly one or more nodes of the WMN having a wired (wireless) connection to the Internet. Thus, the node(s) in the WMN having a wired link to the Internet act as a gateway for traffic originating within the WMN to nodes in the Internet (external to the WMN) and vice versa. Nodes not directly connected to one another in the WMN can communicate with each other using multihop routes, where nodes in the WMN act as routers to route the packets to the intended destination. As in most typical application scenarios we consider the topology of the WMN backbone to be stable and that the nodes are not highly mobile such that normal issues of rapid link breakage and changes in the neighbourhood of nodes, which are induced by node mobility are not an issue in these WMNs.

Fig. 1.2 shows a typical deployment scenario for a wireless mesh network. As seen from the figure the wireless mesh nodes which are usually static collaborate to form a wireless network topology which can then be used by the mobile nodes too by associating with some of the wireless mesh nodes which are in their neighbourhood. One or more nodes forming the wireless mesh may have access to the external Internet and other networks via wired (wireless) links. Such nodes form a gateway for traffic originating at nodes external to the wireless mesh

and having destinations served by the wireless mesh. Note, here, when mobile nodes associate with the wireless mesh backbone, their traffic can be assumed to be originating at the wireless mesh node serving these nodes, and traffic going to these nodes can be assumed to be destined towards the wireless mesh node serving the mobile nodes. In a typical scenario we will consider a single node to be the gateway for nodes in and served by the wireless mesh network to access the Internet. Nodes in or served by the wireless mesh network can communicate with one another as well as with external nodes via multihop routes. Thus, the wireless mesh nodes collaborate to route and forward packets on behalf of other nodes in the network.

For the purpose of our thesis, we limit our focus only to the wireless mesh backbone formed by the wireless mesh links as shown in Fig. 1.2. Traffic, from and to external nodes or networks, will be assumed to originate at or be destined for the wireless mesh node which forms the ingress or exit points for such traffic respectively. Further, we assume that the nodes forming the backbone wireless mesh network use omnidirectional antennae and schedule transmissions on a single fixed channel.

Reservation based WMNs supporting QoS are ideal for provider managed WMNs where customers expect QoS for multimedia traffic. A significant portion of traffic in such WMNs will be composed of unicast traffic flows between communicating peers, e.g. Voice over IP (VoIP), File Transfer Protocol (FTP), Internet browsing sessions, P2P Traffic, and different kinds of rich multimedia traffic (see Refs. [108, 109] for more detailed discussions on multimedia and Peer-to-Peer (P2P) traffic in networks). Thus, for the investigation in this thesis we assume a deployment scenario where we have multiple unicast flows in the WMN. Network operators, however, are interested in not only supporting QoS, but also in maximizing the throughput of their wireless mesh networks. Network coding has recently been identified as a promising means to obtain bandwidth savings in WMNs, which, in turn, can translate to higher throughput in the WMN. However, to date, network coding has been studied within the context of WMNs mainly using contention based means such as Carrier Sense Multiple Access/Collision Avoidance (CSMA/CA) to access the wireless medium. In fact, most of the work studying network coding in WMNs is limited to work carried out in WMNs based on the IEEE 802.11 [41] standard.

However, the network coding solutions designed and tested in WMNs so far make implicit assumptions about the medium access control regulating schemes. Reservation based MACs for WMNs present a radical change to medium access in WMNs and hence we need to investigate in depth network coding solutions for reservation based WMNs if one is to realize the true benefit of network coding in such networks. Firstly, the contemporary network coding schemes do not consider the overhead of bandwidth reservation and coordination for scheduling broadcasts in reservation based WMNs. They rely on the inherently broadcast nature of the wireless medium, which in a reservation based WMN needs to be accessed in a scheduled manner. This leads to inefficient performance of the contemporary network coding schemes when applied to reservation based WMNs. An efficient approach to deployment of network coding in reservation based WMNs is missing.

In this thesis we aim to address the above gap and look at mechanisms to practically and efficiently deploy network coding in reservation based WMNs to achieve increased throughput. This will enable network operators managing such WMNs to support QoS for individual traffic flows in the WMN using the good support for QoS offered by reservation based MACs and at the same time increase the throughput in their networks. In particular, in this thesis, we will look at WMNs using TDMA/TDD based bandwidth reservation schemes. To enable us to put our research in concrete settings and demonstrate a proof of concept for the solutions developed in this thesis, we will develop and present the solutions within the concrete settings of the IEEE 802.16 Mesh mode. The scientific concepts investigated and the solutions developed are,

however, of a more generic nature, and can be transferred to reservation based WMNs using other technologies too.

In the next section we will outline the concrete research contributions presented in this thesis.

1.2 Contributions

As outlined previously, the main goal of the work presented in this thesis is to investigate and design mechanisms for effectively deploying network coding in WMNs which use reservation based MAC protocols. In particular the goals and contributions of the thesis are as follows:

- We investigate the implications of using contemporary approaches to network coding in reservation based WMNs. The developed analytical model permits us to gain deeper insights into the pitfalls and issues involved when deploying network coding in reservation based WMNs. This allows us to derive design principles for network coding solutions in such WMNs.
- Based on the previous analysis and design principles, we next develop and design distributed mechanisms for enabling network coding to be deployed efficiently in reservation based WMNs. In particular, we move away from the opportunistic, packet oriented[†] network coding solutions found in the literature, and propose and design solutions for network coding which are stream oriented. The solutions designed for stream-oriented network coding are distributed in nature, and require only information locally available at nodes, or readily obtainable from the neighbouring nodes. We thus present a new paradigm for stream-oriented network coding.
- An optimal deployment of network coding in the entire WMN, however, cannot only rely on local distributed solutions. Hence, we also design and evaluate our framework for network coding termed CORE (Centrally Optimized Routing Extensions). CORE , as the name suggests extends the default routing in the WMN by adapting routes to achieve its optimization goal. The main optimization goals of CORE are to achieve bandwidth savings in the network via adapting the routes to minimize interference, and also increase the number of network coding opportunities in the WMN. CORE relies on a cross-layer approach whereby it uses information obtained at the MAC layer as an input for its route adaptation. Further, with the route adaptation, it also interacts with the MAC layer, giving notifications as to the bandwidth reservations to be carried out at the MAC layer.

1.3 Outline of the Document

In this section we briefly outline the structure of the rest of this thesis to enable readers to easily find their bearings when reading this work.

In Chap. 2 we will look at work from the literature which is closely related to the work in this thesis, as well as provide an introduction to the fundamental concepts from related work which are considered in this thesis. We use the IEEE 802.16 standard's MeSH mode as a prototype for reservation based MACs to test our developed algorithms and provide a proof-of-concept for the developed methods. Hence, in Chap. 3 we present the details of the scheduling mechanisms provided by the IEEE 802.16 MeSH mode with a focus on distributed scheduling mechanisms. This brings us to the end of Part I of this thesis whose aim is to provide a motivation for the work, and outline related work to permit the readers to see this work in perspective.

[†] More details about contemporary approaches to network coding are presented in Chap. 2

In the next part of the thesis, i.e. Part II, we start looking at the concrete problem being addressed in this thesis — namely, practical and efficient network coding solutions for reservation based WMNs — in detail. Initially, in Chap. 4 we look at the issues involved in deploying contemporary network coding solutions to reservation based WMNs. Here, we look at the problem analytically, and identify the pitfalls when one uses the current network coding solutions in reservation based WMNs. This leads us to derive design principles which provide guidelines when designing network coding solutions for WMNs using more sophisticated reservation based MAC layers. Our design principles lead us to propose and put forward the notion of using Stream-Oriented Network Coding (SONC) in reservation based WMNs and move away from conventional opportunistic packet-by-packet network coding decisions. In the next chapter (Chap. 5), we use the insights obtained from the analysis in Chap. 4 to develop network coding solutions for reservation based WMNs. In particular, we look into distributed solutions for detection of network coding opportunities at the stream-level. We also design and present protocols for advanced scheduling to enable use of network coding once suitable network coding opportunities are detected. Within the scope of the latter study we also design and propose schemes to measure the degree of benefit which can be obtained by using network coding in WMN using scheduled transmissions. This enables an estimation of the benefit which one can expect if a detected network coding opportunity is actually realized using scheduled network coding transmissions. In particular, the metrics developed thereby are light-weight and do not require complete knowledge of network wide transmission schedules. This is followed by an evaluation presented in Chap. 6 to demonstrate proof-of-concept for the solutions developed so far. In particular we note that simple and distributed solutions for stream-oriented network coding as presented are quite robust and are able to effectively reduce the bandwidth needed for transmissions in reservation based WMNs.

However, if one looks at the routing, scheduling, and network coding across the entire WMN as a joint optimization problem, further gains can be obtained. This is the basis for the research presented in the next part of the thesis (Part. III). In Part III we look at solutions for network coding in reservation based WMNs which span the entire WMN. In particular we present our solution Centrally Optimized Routing Extensions (CORE) which enables more gain via use of network coding in reservation based WMNs. CORE, as the name suggests, is a centralized solution, which extends existing routing solutions to adapt the routes in the WMN to enable increased deployment of network coding in the WMN. CORE, uses efficient heuristics which permit operation in near-real-time producing solutions within an operator specified bound on the computation. Although, CORE uses a central server, it uses distributed means to execute the centrally computed solution. Hence, it is able to fit well with and use the distributed solutions presented in Part II of the thesis. This, also makes the WMN more robust to failures of the central server, and also permits normal operation of the WMN without the presence of a central server. Moreover, the design choices made reduce the amount of information which needs to be collected centrally to compute optimal solutions for network coding at the scale of the entire WMN[‡]. Thus, in Part III we first present and overview of the CORE framework in Chap 7, followed by the details of the working of the individual CORE components in Chap. 8. We then go on to present an evaluation of the developed global solutions in Chap. 9.

We then come to the last part of this thesis, namely, Part IV. In this part we draw up conclusions from the study presented within this thesis, and identify avenues for future research.

The appendices following this part of the thesis present additional details and solutions developed which though vital and important are not central to conveying the overall research findings presented in the thesis.

[‡] This is termed as a global scale for us as we restrict our discussion and research only to the WMN as already mentioned in Sec. 1.1

2 Related Work

The goal of this chapter is to introduce in brief the readers of this thesis to the work in the literature which is closely related to the work in this thesis as well as associated concepts.

Bandwidth is a scarce resource in wireless networks and needs to be efficiently and optimally utilized. This is especially the case for wireless multihop networks such as mesh networks. Multihop mesh networks are promising means to extend the coverage of existing cellular wireless networks [64] as well as to serve for building a wireless backbone for community wide or enterprise wide networks (for e.g. [71, 103]). Wireless multihop networks which form a network infrastructure and are composed of static nodes are classified as *Infrastructure/Backbone* WMNs in Ref. [4]. In such networks dedicated nodes called mesh routers form a network architecture for clients. The end client nodes are not involved in activities such as routing or scheduling. A selected number of the mesh routers may provide gateway functionality to the other mesh routers and the clients in turn. Such gateways connect the WMN to the Internet or to other networks allowing data transfer between the clients and nodes in external networks.

Multihop wireless networks can also help facilitate communication in the absence of fixed infrastructure (also called ad hoc networks). Ref. [4] classifies such networks as *Client WMNs*. Here, the client nodes also need to perform routing functionality and the network topology is more unstable with nodes periodically joining and leaving the network. A third category of wireless multihop networks as classified in Ref. [4] is that of *Hybrid WMNs*. Hybrid WMNs are a combination of Infrastructure WMNs and Client WMNs.

The focus of this thesis is on Infrastructure WMNs which are usually managed by a network operator, and are assumed to be stable with static topology. We are moreover interested in WMNs which can support strict QoS guarantees (see Refs. [102, 35] for a discussion of different QoS architectures). An example of such a WMN standard is the IEEE 802.16 MeSH mode (we present detailed insights into this in Chap. 3 and in Ref. [83]). Bandwidth still remains a valuable resource in such networks, and network providers usually seek to increase the traffic carrying capacity of their WMNs. For wireless networks in particular breaking away from the strict layered approach of the Internet protocol stack brings additional benefits. Cross-Layer Optimization has been proposed as a means to optimize the performance in WMNs. We will discuss this in brief next as this is one of the building blocks of this thesis.

2.1 Cross-Layer Optimization

As discussed in Ref. [106] cross-layer design has been seen as a promising option to increased efficiency of operation in WMNs. There the authors discuss different definitions of cross-layer design. In a strictly layered protocol stack interaction is limited to adjacent layers and the decisions made by one layer are independent of the decisions made by the other adjacent as well as non-adjacent layers. The services provided by the individual layers in the protocol stack are realized by protocols designed independently for the different layers.

So one could term protocol design which violates the reference layered protocol stack as a cross-layer design. Such violations may be exposed by interaction among non-adjacent layers, or sharing state information internal to one layer with services at other layers. Such a cross-layer design is especially of interest in wireless networks. Some of the prior approaches suggest

a completely new design and moving away from the layered architecture (e.g. Ref. [12]). The major motivations for enabling cross-layer information flow, as pointed out in Ref. [106], are the unique problems and characteristics of the wireless medium, which at the same time offers new modalities of communication, in addition to the possibility of opportunistic communication in such networks which is not present in traditional wired networks. Transmission Control Protocol [95, 5] (TCP) is known to perform badly in wireless networks (see references [118], [117] for a discussion). This is mainly because the congestion control algorithms from TCP [110, 5] were originally designed to consider packet loss as a sign of congestion in the network and react accordingly. However, in wireless networks the data loss rate due to interference and noise may be much higher than packet drops or loss due to congestion. This leads to TCP showing poor performance as the protocol at the transport layer operates without information from the medium access control layer and the current channel conditions. Recognizing this permits one to improve the performance of TCP in wireless networks (e.g. Ref. [8, 105]). The works [7, 6] presents a survey of some TCP specific improvements using cross-layer design.

Similarly cross-layer design can also be used to better utilize the wireless channel [96, 105]. For example Sadeghi et al. in Ref. [99, 98] present OAR, which is an opportunistic medium access protocol for ad hoc wireless networks. Using OAR nodes in the network try to exploit good channel conditions by sending multiple back-to-back data packets, thereby optimally utilizing good channel conditions. Such mechanisms would not be possible if the Medium Access Control (MAC) layer would not consider Physical (PHY) layer information. Similar arguments support the cross-layer design in Ref. [46] where the work in [99, 98] is extended to the case of multiple channels. Routing algorithms can also profit in the choice of paths by using information from the lower layers. For example, in Ref. [21] the authors present the routing metric ETX (short for expected transmission count) for routing in wireless ad hoc networks. The ETX metric gives the number of transmissions needed on a link to successfully transmit a given packet under the current channel conditions. Their evaluation shows that using the ETX metric improves the throughput significantly especially for routes with more than two hops. The authors in Ref. [30] use link quality measurements to switch routing preemptively (i.e. prior to route break) by switching to links with better link quality. There the link quality is assumed to be a function of measured signal strength. Similar approaches which use thresholds for link quality to avoid using certain links for routing can be found in [16, 26, 69].

The work [25] provides a good comparison of routing metrics for static multihop wireless networks, i.e. WMNs which are the focus of this thesis. The work finds out that for static and stable wireless multihop networks a routing algorithm can select better paths by explicitly considering the quality of the links in the route. However, their findings show that even with slight mobility of the final end users (i.e. say the source or destination), metrics which depend on measuring link quality perform quite poorly. The reason as mentioned in [25] is that under dynamic conditions the link quality measurements are of not sufficient good quality and require time to come up with stable measurements. Simple hop count metric on the other hand provides a good performance even under such conditions. Hence, in our thesis we use the simple hop count metric for routing as a benchmark to compare our results as it is not affected by the peculiarities of the more dynamic routing metrics.

Thus, in general, we see that there is a vast amount of prior work on cross-layer design. However, most of the work has been carried out in the scope of the IEEE 802.11 standard and substantial work in the area of multihop reservation based WMNs as targeted by this thesis is still missing. Cross-layer design is however not without its own complications. The work [105] points out the potential benefits and at the same time points out to the major research challenges which need to be tackled to fully profit from cross-layer design. The authors in Ref. [50] go a step further and suggest that caution should be exercised when deploying cross-layer design in

wireless networks. They identify that a modular architecture as provided by the strictly layered Internet protocol stack as well as the OSI models has been of fundamental importance in the success of the Internet. This architecture has permitted protocol designers working at one layer of protocols to largely ignore the rest of the protocol stack. This, is however, not possible once the strict separation of the layers is broken down via cross-layer design. The interaction between layers may lead to unforeseen dependencies and lead to disastrous overall performance. One case study presented there is of a rate-adaptive MAC protocol [39]. The authors in Ref. [50] point out that when such a MAC is used with minimum-hop routing (which is the default case for most networks), the performance drops. The reason being that the shortest-path routing tends to choose longer hops for which the signal strength is lower, which in turn leads the rate-adaptive MAC layer to further reduce the data rate. Hence, in general caution is recommended when designing cross-layer optimized solutions for wireless network. The works [50] and [3] provide a set of general guidelines for cross-layer design. These recommend that cross-layer design should be employed only if the expected gains are sufficient to offset the added complexity. Thus, one should try to stay with a layered protocol model as far as possible and derive as much benefits as possible using layered solutions. Further, they recommend that cross-layer design should not violate requirements of standards and should be possible to implement in using standardized protocols. Dependency between protocol stacks should be kept to a minimum, and where it exists it should be carefully evaluated. One should also avoid that a single parameter is acted on by multiple layers at the same time scale. Finally unbridled cross-layer design should be avoided.

In this thesis we use cross-layer design as we expect to achieve significant performance gains which are not otherwise obtainable. Additionally we keep to a loosely coupled cross-layer design where we optimize individual layers using information from other layers and do not optimize the layers together. This makes it easier for the solutions developed at the individual layers to work with other protocols. Additionally our solutions are implemented within the framework of a standardized protocol (IEEE 802.16-2004 [42]) which shows that the solutions are implementable and interoperable with standards which do not use a cross-layer design approach. To avoid unbridled cross-layer design we introduce a conceptual cross-layer database which holds and maintains the variables which can be accessed by different layers and also provides additional functions to individual layers using information from other layers (see Fig. 8.3). This approach is similar to the shared database architectural blueprint identified for cross-layer design in [106]. Our solution aims to allow us to simultaneously optimize the performance of routing, scheduling as well as use network coding efficiently in reservation based WMNs. The goal being to improve the traffic carrying capacity of the WMN as a whole.

Routing and scheduling cannot be separately considered in WMNs if a good performance is to be achieved. For instance if the individual flows in a WMN are assigned routes without consideration of the feasibility of a schedule for their demands then very high delays and backlogged traffic will be the result. In the next section we will look at some of the related work which looks into the aspects of routing in WMNs and also looks into the scheduling aspects in parallel.

2.2 Routing and Scheduling in Wireless Mesh Networks

Routing and scheduling in wireless networks have both been studied to quite some extent. Ref. [11] provides an early overview of routing protocols designed especially for ad hoc networks. Here the authors classify the routing protocols into two categories, firstly table driven or proactive protocols and secondly on demand or source-initiated routing protocols. Destination-Sequenced Distance-Vector Routing (DSDV) [94] and Wireless Routing Protocol (WRP) [86] are some examples of table driven routing protocols. Dynamic Source Routing (DSR) [45], Ad

hoc On Demand Distance Vector Routing (AODV) [92, 93] and Signal Stability-Based Adaptive Routing (SSA) [26] being some representatives of the second category.

Similarly there exists a lot of work in the area of scheduling in wireless networks. In multihop wireless mesh networks in which we are interested our focus mainly lies on Spatial TDMA (STDMA). Ref. [89] is one of the first works to look into STDMA for wireless multihop networks. STDMA is a generalized case of TDMA where nodes in different spatial zones of the wireless network are permitted to schedule transmissions provided that these do not interfere with each other given a model for interference. The authors in [17] look into distributed solutions for dynamic assignment of channels to nodes in a multihop packet radio network. Here, they split the channel into a control segment and a data transmission segment, with the control segment being used to avoid data transmission conflicts. The IEEE 802.16 standard's mesh mode uses similar concepts with mechanisms provided for enabling access to both the control subframe as well as the data subframe in a conflict free manner (see Chap. 3 for more details). Similar to the above works, the seminal work of Tassiulas and Ephremides [113] defines the concept of an activation set which defines a set of servers (nodes in the network) which may be activated in the same slot (i.e. can transmit in the same slot) without violation of the collision-freedom. The constraint set for such a network can then effectively be modelled by a set consisting of all permissible activation vectors of the system. A schedule then simply defines at each time slot t the set of links which are activated.

Routing and scheduling have been recognized to be primary factors in the performance obtained in WMNs and wireless multihop networks in general. This has led to various efforts to look at these two either jointly or in a closely coupled manner (e.g. [57, 62, 61, 53, 97]). However most of the related work in this are either permits fractional scheduling of links or does not consider multihop networks, or use central scheduling schemes. Where distributed schemes are considered no concrete standards are used to validate the solutions (e.g. [61]), and are restricted usually to frame-by-frame computation of schedules. To the best of our knowledge no detailed study can be found in the literature which studies means to fully utilize the sophisticated bandwidth reservation possibilities provided by the IEEE 802.16 MeSH mode and similar systems.

In addition to the above recently network coding has been proposed as a novel means to improve the traffic carrying capacity of wireless mesh networks. It is the goal of this thesis to use network coding effectively in reservation based WMNs. We give an overview of network coding and related work in Sec. 2.3.

2.3 Network Coding

Work in the area of network coding was initiated by the seminal work of Ahlswede et al. [2]. There the authors showed that permitting routers to combine information from different packets inside the network and then forwarding such coded packets permits the network to achieve multicast capacity. To explain the idea let us look at the example in Fig. 2.1.

Consider the simple wired network topology shown in Fig. 2.1(a). As shown in the figure each link is assumed to have a capacity to transmit one bit at a given instant in time. Assume we have a single source s and two destination nodes (p and q), i.e. we have a multicast transmission. It can be shown (see [2]) that the multicast capacity for the network is two bits per unit of time. However, using routing where received packets are simply forwarded we cannot achieve this multicast capacity of the network. On the other hand permitting nodes in the network to mix information from different packets together before forwarding them can allow the multicast capacity to be reached.

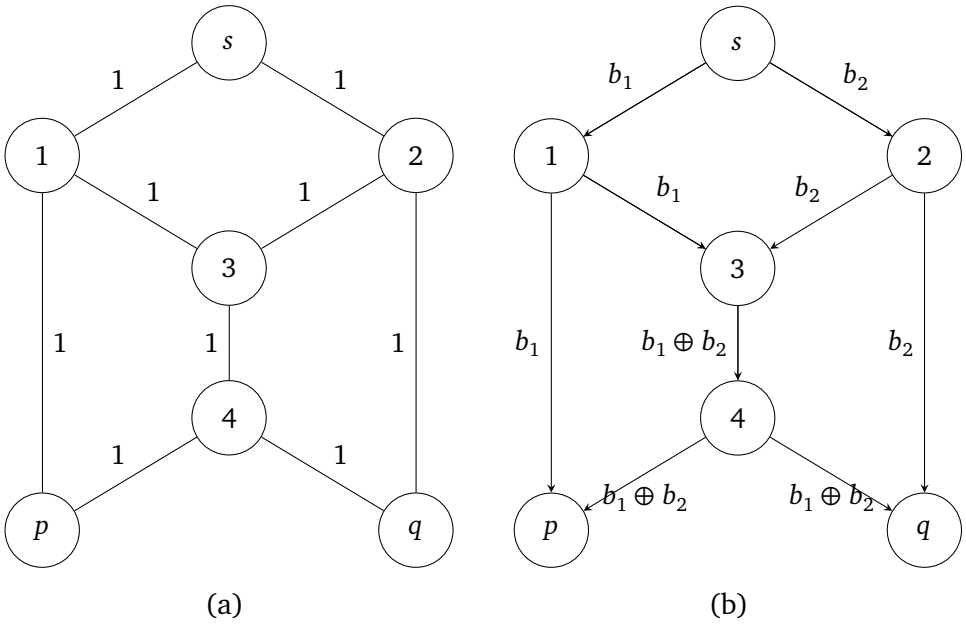


Figure 2.1: Simple Example to Demonstrate the Principle of Network Coding [2]

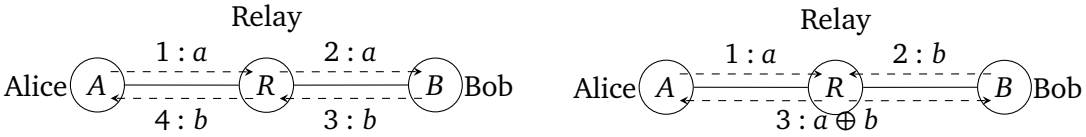
Network coding permits an intermediate node in the network to combine a number of packets it has received or generated itself to create one or more outgoing packets which are then forwarded to the other nodes in the network. Thus, using network coding a node does not as a simple relay for packets repeating them on their way to the destination. Network coding by definition subsumes routing, as routing can be considered to be a special case of network coding where the packet is combined with a null packet or with no packet to create the outgoing packet(s).

Fig. 2.1(b) shows such an example of network coding. Here the source node has two bits b_1 and b_2 of information to transmit to the nodes p and q. Here, the node 3 will combine the incoming bits b_1 and b_2 using a XOR operation to form the packet $b_1 \oplus b_2$ which is then transmitted on the outgoing link $(3,4)$. The node 4 will relay this packet as shown to the nodes p and q. Now, each of the nodes p and q will be able to receive correctly the bits b_1 and b_2 from the information reaching them at every time instant. For example at node p we have already received the bit b_1 over the link $(1,p)$. After receiving the packet $b_1 \oplus b_2$ node p can use the operation $b_1 \oplus (b_1 \oplus b_2)$ to correctly also obtain the bit b_2. Similarly, the node q is able to correctly receive both the bits b_1 and b_2. Thus, we can see that network coding permits us to improve the throughput in the network and achieve multicast capacity which cannot be reached using simple routing. As seen from the example we can see that network coding is able to increase the effective information transfer rate beyond the physical capacity (data transfer rate) which is provided by the network. Initial work in the area of network coding focussed mainly on multicast transmissions and wired networks ([54, 58, 38, 68]).

Later work has also looked at network coding in the context of wireless networks ([22, 100]). However, most of the above work has been only theoretical in nature and restricted to the case of application of network coding to multicast traffic. There has also been some work looking into the case of application of network coding to unicast transmissions in networks ([59, 37]). For the case of a single unicast or broadcast session the authors in [60] show that there are no improvements over routing obtainable as far as throughput is concerned when network coding is used. For the case of multiple unicast transmissions the authors in [59] show that network

coding can be beneficial compared to routing. However, the benefit is strongly dependent on the network model and the authors [59] conjecture that for undirected networks with fractional routing the potential for network coding to increase bandwidth efficiency does not exist. For the multiple unicast transmissions case, the authors in [67] show that network coding does not provide any order difference improvement over the bound provided by the work [32] which does not consider network coding. Similar results are presented by [47] for the case of multicast traffic. However, in [66] and [65] the authors show that network coding does bring a constant factor gain over routing for the multiple unicast as well as multicast traffic cases in wireless networks. This gain is very valuable in practice as it means a direct increase in the traffic which can be supported by the network.

Most of the above work discussed is however mainly theoretical in nature. Recently the work of Katti et al. [49, 48] proposed a scheme to practically deploy network coding in wireless mesh networks. This work has been the starting point for most of the practical work on network coding in wireless mesh networks. Our network coding solution also looks at practical deployment issues and is inspired by this work. The authors in [49, 48] use a simple XOR coding scheme and require that the coded packets be decoded by the immediate next hop nodes. The reason being to avoid transmission of unnecessary data to parts of the WMN where it may not be needed. Further this permits one to keep the complexity of the coding and decoding to a minimum. These requirements are also important for our scenario and hence we also use such a simple XOR coding scheme for our work.



(a) Without Network Coding (b) With Network Coding

Figure 2.2: Simple Alice-Relay-Bob Example Showing Benefit of Network Coding

Consider the example in Fig. 2.2. We have the wireless network topology as shown in the figure with two participants Alice and Bob wanting to exchange messages. Alice wants to send message a to Bob, and Bob wants to send message b to Alice. Let us assume that the messages are of the same size. As the topology does not permit direct communication between these two communication partners their messages have to be relayed by the relay node R. Fig. 2.2(a) shows how the message exchange takes place with standard routing. In time slot 1 the node A relays packet a from Alice to the relay R. In slot number 2 the relay node will then repeat the packet transmission thereby forwarding packet a to the node B and thus to Bob. Similarly, the message b from Bob will be relayed in time slots 3, and 4 as shown in Fig. 2.2(a). We see that we need a total of 4 transmissions (4 time slots) for the message exchange.

Fig. 2.2(b) shows the same message exchange using COPE (the network coding scheme proposed in [49, 48]). Here, the message a will be relayed from node A to node R in time slot 1. The message b will be relayed from node B to node R in time slot 2. After obtaining both the messages the relay node R in time slot 3 transmits the packet $a \oplus b$ as a multicast transmission to nodes A and B. Where $a \oplus b$ is obtained by XORing the contents of packets a and b. Node A is then able to obtain the packet b by performing the operation $a \oplus (a \oplus b)$ as it has already the packet a (sent by itself). Similarly node B is able to successfully recover the message a by the operation $b \oplus (a \oplus b)$. We can see that even in this simple example using COPE we can save one fourth of the bandwidth which was required for the message exchange without using network coding. The authors in [49, 48] define the coding gain as the ratio of the transmissions required

without coding to the transmissions required using COPE. They show that using COPE considerable gains can be obtained. However, the metric used to measure gain is not suitable for our reservation based scenario (see Sec. 5.3.2 and Appendix A), and neither is the application of COPE efficient in the reservation based WMNs we are targetting (see Sec. 4.1).

Network coding is thus in general valuable for wireless networks. However, as shown in Refs. [101, 51] other issues such as routing, topology, and interference are to be considered too. In the next section we will briefly look at the related work considering routing, scheduling as well as network coding in wireless networks.

2.4 Joint Routing, Scheduling and Network Coding

Refs. [75, 104] are one of the first works looking into the issue of considering routing, scheduling and network coding jointly in wireless mesh networks. The authors in [104] build up on the work in COPE [49] by considering network coding and routing as a joint problem. The authors in [104] use a linear programming based approach to find an optimal solution in an 802.11 based MAC. However, the solution presented requires multipath routing and fractional bandwidth allocation on links. Further, the authors do not present any protocol to implement the presented solution in real WMNs. In particular, for reasons of avoiding jitter and out of order packet delivery issues we do not want to split packets belonging to a single flow along multiple paths. Additionally, it is not possible for us to reserve, e.g., 0.333 of the link capacity for a particular link in the WMN (see discussion in Chap. 3). Only an integer number of minislots may be reserved for transmissions on a link. The above constraints put our problem in the class of Integer Linear Programming (ILP) problems, which are generally considered to be NP-hard [18].

We modeled our problem (see Sec. 8.1 for further discussion) as an ILP using the *GNU Linear Programming Toolkit* [29]. This open-source toolkit is able to solve ILP problems efficiently using a *branch-and-bound* approach. For small networks (7–10 nodes, 10–15 links, 2–5 flows) and very few (1-10) minislots, the ILP was solved within a few tens of seconds with reasonable results: correct routes where found and no collisions occurred. For larger networks (e.g. 16 nodes in a 4x4 grid layout) the solver was not able to find a solution in reasonable time (24 hours); an increase in flows and/or links had similar effects on the solution time.

Almost all of the work in the field of network coding assumes a 802.11 or similar MAC. No study is made for a reservation based MAC like the MeSH mode. An important issue when using network coding is that it should not add a high delay penalty for the packets when coding is used (see [49] for arguments). Prior literature on wireless network coding makes coding decisions mainly on a packet by packet basis, which is not feasible without high delay in the MeSH mode (see Sec. 4.1 for a more detailed discussion). Reserving multicast bandwidth on a packet by packet basis via the three-way handshake in the MeSH mode is not only difficult (the receiving nodes have to agree to grant the same set of slots) but also involves the three-way handshake delay which can be considerable (see [14]), especially for multicast reservations.

Most of the other literature on scheduling and network coding is either restricted to multicast traffic (e.g. [100]), or does not permit spatial reuse for scheduling as specified for the MeSH mode (e.g. [15]). Other work such as [101] also consider joint scheduling and network coding, but do not consider in detail the coordination overhead for reservation based WMNs.

We thus conclude that routing, scheduling and network coding need to be considered jointly in wireless mesh networks. However, there is still a lack of a detailed study of practical solutions for sophisticated reservation based WMNs. This thesis aims to bridge this gap.

2.5 Summary

In this chapter we have provided a short overview of the most relevant related work as well as provided an introduction to the basic concepts used in this thesis. In the next chapter we will continue our survey of the background needed for this thesis and provide details of the IEEE 802.16 MeSH mode which we use as a concrete setting to provide a proof-of-concept for our solutions.

3 Scheduling in the IEEE 802.16 MeSH Mode

As discussed earlier, we will be using the IEEE 802.16 standard's MeSH mode as a prototype MAC layer providing sophisticated TDMA/TDD based bandwidth reservation features. This enables us not only to demonstrate the proof-of-concept for the research but also enables us to validate the research in more concrete settings. In this chapter we provide the readers with an overview of the relevant parts of the standard's MeSH mode which are necessary to understand the later chapters. Further details of the standard can be found in [42]. Ref. [83] provides a comprehensive overview of the MeSH mode of the IEEE 802.16 standard.

3.1 Overview of the IEEE 802.16 Standard

Fig. 3.1 shows the scope of the IEEE 802.16 standard [42]. As seen from the figure, the IEEE 802.16 standard specifications cover the MAC and Physical (PHY) layers of the Open Systems Interconnect (OSI) protocol layer model (see [112] for a detailed overview of the OSI stack).

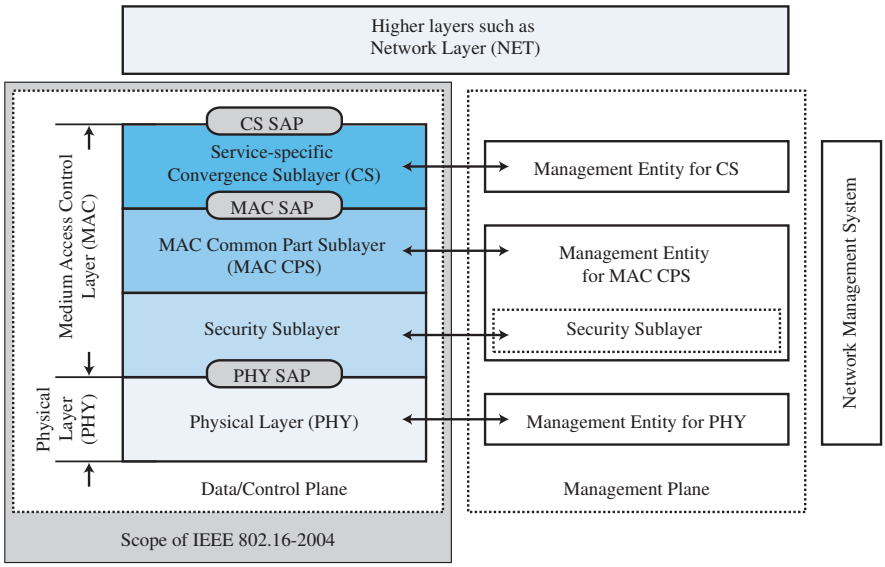


Figure 3.1: Scope of the IEEE 802.16 Standard

The standard provides the primitives for accessing the Service Access Points (SAP) to obtain services from the upper and lower sublayers. It also defines formats for the messages which are used by the standard. However, the standard completely leaves the choice of management plane functions open to permit vendor optimization. Thus, although we use the IEEE 802.16 standard as a prototype for our research we have had to design, develop and implement a significant amount of management plane functions before we could start with the research presented here (see for e.g. the references [81, 82, 83, 77, 85, 84, 79, 74, 75, 31] for more details about some work done within the scope of the thesis for the IEEE 802.16 functionality). The mechanisms at the PHY layer are out of scope of this thesis and are not of much importance to understand the work. Hence, in this chapter we limit the discussion mainly to the MAC Common Part Sublayer (MAC CPS) which provides the vital functionality of bandwidth reservation and scheduling.

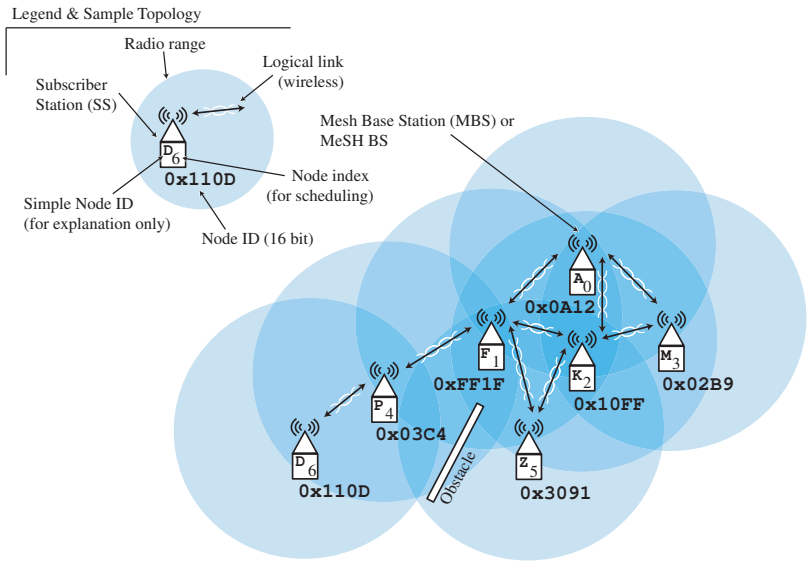


Figure 3.2: Sample Topology for the MeSH Mode

The 802.16 standard provides different modes of operation, namely the PMP mode of operation, the recently introduced relay mode of operation, and the MeSH mode of operation. In the PMP mode of operation all the subscriber stations have to be within radio range of the base station, and thus the network is limited to a strictly star topology. The relay mode of operation of the standard introduces in addition to base stations so called Relay Stations (RS) which enable extended range and/or throughput improvement. Subscriber stations no longer need to be within direct range of the base station but may communicate with the base station via relay stations which help transmit packets between the base station and the subscriber stations over multihop paths. However, although it permits multihop communication, in the relay mode the network topology is still limited to a tree topology. Thus, the above modes of operation of the standard are not suitable for wireless mesh networks. The MeSH mode of operation of the IEEE 802.16 standard on the other hand permits a full fledged mesh topology and defines both centralized and distributed schemes for managing bandwidth reservations in the WMN. When the bandwidth reservations are managed centrally (by a Mesh Base Station (MBS)) the scheduling is referred to as Centralized Scheduling (CSCH). However, CSCH is also limited to bandwidth management on a tree rooted at the MBS. The MeSH mode also specifies a distributed mode of bandwidth reservation termed Distributed Scheduling (DSCH). Distributed scheduling is more flexible than CSCH and can be used to reserve bandwidth and schedule transmissions on all the links in the WMN. Hence, for this work we will focus on DSCH and assume, without loss of generality, that it is the only means of bandwidth reservation in the WMN. An overview of the IEEE 802.16 specifications can be found in [27].

Fig. 3.2 provides a sample topology for a WMN using the MeSH mode. As shown in the figure a single node is chosen as the base station (MBS) for the entire WMN. It is usually the node which provides the rest of the WMN with access to external networks and the Internet (possibly using a separate wired link). New nodes join the WMN by using a network entry process using a neighbouring node which is already in the WMN as a sponsor for the network entry (see Ref. [42] for details). The MBS is responsible for maintaining and advertising the network configuration parameters (e.g. frame lengths, modulation etc.). It is also responsible for authentication of new SSs wanting to join the WMN as well as admission control for flows entering the network. The MAC Security Sublayer provides functionality for authentication for new nodes entering the network and per-link data encryption for the data transmissions when the network is in operation. The security sublayer functions are not the focus of this work.

From Fig. 3.2 we see that each node has a 16 bit unique Node ID which is used to address the node in the WMN. This node identifier is obtained by the node from the MBS during the network entry process. Nodes connect to direct neighbours which are within radio range. Additionally, each node as well as the MBS is responsible for periodically transmitting network configuration messages in the control subframe (explained later) to maintain the network. Using the information contained in these messages nodes keep track of the nodes in their two-hop neighbourhood and also their individual node identifiers. Furthermore, using the different types of control messages (e.g. schedule control messages) transmitted in their neighbourhoods nodes also maintain a record of the data and control transmission schedules of the nodes in their neighbourhood.

3.2 Frame Structure for the MeSH Mode of the IEEE 802.16 Standard

As mentioned previously the MeSH mode uses TDMA/TDD based bandwidth reservation scheme to schedule transmissions in the WMN. This enables the WMN to control the bandwidth allocation and QoS on individual links at a finer granularity than the IEEE 802.11 based WMNs.

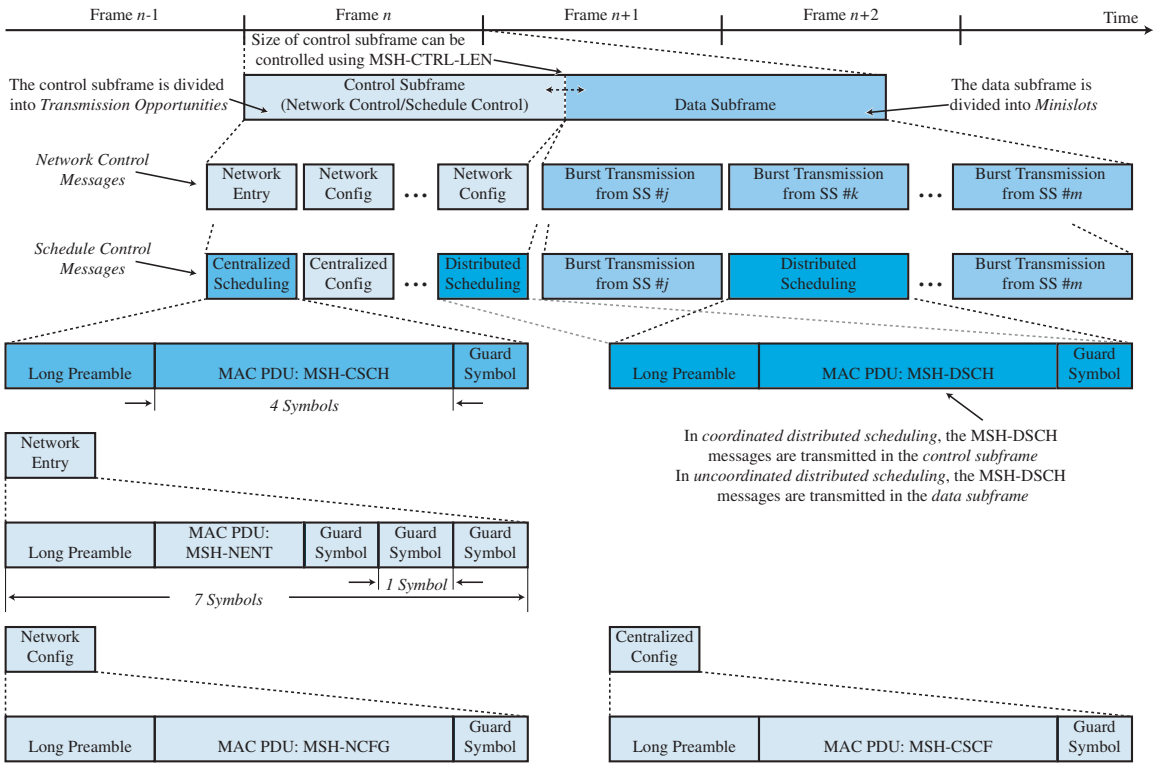


Figure 3.3: Mesh Frame Structure and Messages

Fig. 3.3 shows the frame structure used in the MeSH mode. Time is divided into frames, where each frame is composed of two subframes, one control subframe and one data subframe. There are two types of control subframes, network control subframes and schedule control subframes. Network control subframes are used for transmitting messages relevant to maintenance of the network. Schedule control subframes are used for transmitting messages for coordinating the data transmission schedules of the nodes in the network via bandwidth reservations. These two types of control subframes alternate with each other, whereby the number of schedule control subframes is much more than the number of network control subframes. E.g. every ninth frame may contain a network control subframe with all the frames in between containing schedule control subframes. Each control subframe is further divided into a number of transmission op-

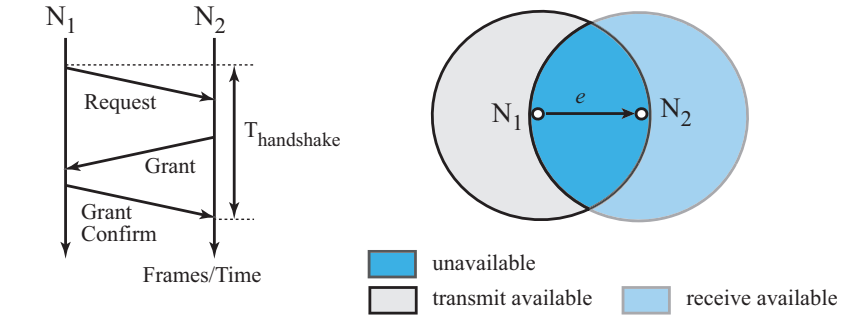
portunities. The first transmission opportunity in a network control subframe is reserved for network entry allowing new nodes to join the network. The remaining transmission opportunities in the network control subframe are used for transmitting Network Configuration (NCFG) messages.

The schedule control subframe is used for transmission of messages for coordinating the data transmission schedules of the nodes in the WMN. The schedules are computed such that the data transmissions are collision free under the given interference model. The schedule control subframe is divided into two parts. The first part of the schedule control subframe is used to transmit messages pertaining to centralized scheduling. The second part of the schedule control subframe is used for transmitting messages for distributed scheduling. Similar to the network control subframe, the schedule control subframe is also composed of the certain number of transmission opportunities. The number of transmission opportunities devoted to centralized and distributed scheduling is decided by the network configuration, and is advertised by the MBS and the other nodes relay these advertisements using the network configuration messages. For further details the readers are referred to [83, 42].

The data subframe is divided into a number of so called minislots. A minislot is the smallest unit of scheduling in the MeSH mode, and thus represents the smallest unit of bandwidth which can be reserved and allocated to individual links in the WMN. Similar to the division of the schedule control subframe, the data subframe is divided into two parts. The first part of the data subframe is used for data transmissions managed by centralized scheduling, the second part is used for scheduling data transmission using distributed scheduling. The division of the data subframe, i.e. how many minislots are allocated to centralized scheduling and how many for distributed scheduling, is decided by the network configuration parameters. Note, the configuration parameters may be such that zero minislots are kept aside for either centralized scheduling or distributed scheduling. In this case the data subframe can be managed only via one of the two scheduling modes. Similarly, for the schedule control subframe the number of transmission opportunities kept aside for centralized scheduling or distributed scheduling may be zero.

3.3 Distributed Scheduling in the IEEE 802.16 MeSH Mode

As discussed previously, we will be focusing on distributed scheduling solely. Hence, we will now give more details about distributed scheduling in the MeSH mode.



(a) Three-way handshake (b) Minislot state during transmission on edge/link e

Figure 3.4: Bandwidth Reservation Via Distributed Scheduling (DSCH)

Consider Fig. 3.4. Say we have the nodes N_1 and N_2 as shown. And node N_1 wants to reserve bandwidth for transmitting data on the link (edge) e. With distributed scheduling, the nodes use a three-way handshake as shown in Fig. 3.4 (a) to reserve the bandwidth for individual links in

the WMN. In our example, N_1 first issues a request for bandwidth, the node N_2 then responds with a suitable grant specifying the minislots granted for the transmission. On reception of the grant, the node N_1 acknowledges and confirms the grant via a grant confirmation. This completes the three-way handshake. After the three-way handshake is complete the node N_1 may use the granted slots for its data transmissions on the corresponding edge. The three-way handshake ensures that the schedules thus computed are collision free. To enable this nodes in the WMN associate with each slot* a scheduling status, which indicates how the slots may be used for scheduling further data transmissions.

Slot Status	Implication for Scheduling
Available (*av*)	Slot may be used for scheduling both transmission as well as reception of data.
Transmit Available (*tav*)	Slot may be used only for scheduling transmission of data.
Receive Available (*rav*)	Slot may be used only for scheduling reception of data.
Unavailable (*uav*)	Slot may not be used for scheduling transmission or reception of data.

Table 3.1: Slot States and their Interpretation

Tab. 3.1 shows the different slot states specified in the standard. The state *av* is the default state of a slot in the WMN when the WMN starts operation. It basically means that the slot may be used by the nodes in the WMN to schedule both data transmission as well as reception of data. Note, the same slot (i.e. slot addressed by the same slot number and frame number) may be assigned different slot states at different nodes in the network. Thus, the slot state is always associated to a particular node and reflects that node's view of the transmissions scheduled in the WMN by itself and in its neighbourhood. Slots with state *tav* may be used only for scheduling transmission of data by the nodes. Slots with state *rav* may be used by the nodes only for scheduling reception of data. Slots with state *uav* may not be used by nodes in the WMN for scheduling any additional transmissions or receptions of data. Nodes locally update the status of the slots considering the scheduling messages they have overheard from neighbouring nodes, or those which they have themselves transmitted.

Consider Fig. 3.4 (b), assume that node N_1 wants to reserve a single minislot in a particular frame for the data transmission on edge e. Also assume that the slot has the status *av* at nodes N_1 and N_2 as well as at all the neighbours of nodes N_1 and N_2. Now, as explained earlier the node N_1 will initiate and use a three-way handshake for reserving the said slot. Thus, during the three-way handshake it will first transmit a request message (detailed overview of the individual messages will be given shortly) which indicates the demand (in this case a single slot) and also indicates to node N_2 the slots which are suitable for the transmission to be scheduled (these slots are sent using so called availabilities, i.e. these slots have status either *tav* or *av*) from the point of view of node N_1. This request message addressed to node N_2 is also received by all the neighbouring nodes of N_1. As a response to the request, node N_2 first looks up the status for the slots specified by the availabilities sent with the request to see which of those slots are suitable for scheduling the data reception (i.e. slots with status *rav* or *av*). A subset of these mutually suitable slots is then chosen by N_2 as the slots to be granted. This choice is then indicated by N_2 to node N_1 by transmitting a grant message. This grant message is received by N_1 and also by all the neighbours of node N_2. In our example this will lead to all the nodes hearing the grant to update their slot state for the granted slot such that no transmission of data can be scheduled

* We will use the terms minislot and slot interchangeably where it is clear from the context that we are referring to minislots in the data subframe.

simultaneously. On reception of the grant node N_1 will check if the granted slot is still suitable for transmission (based on the slot status) and if this is the case N_1 will send a confirmation for the grant. Node N_2 and all the neighbouring nodes of N_1 will receive the grant confirmation. On reception of the grant confirmation the nodes overhearing this grant confirm will update the status for the confirmed slots such that no reception of data can be scheduled for these slots. In our simple example, where prior to the handshake all the nodes had slot state available for the slot being reserved, the nodes will have slot states as shown in Fig. 3.4 (b) at the end of the handshake. Thus, the working of the three-way handshake can be considered to be roughly similar to the Request To Send/Clear To Send mechanism used by IEEE 802.11 based networks. The major difference here is the way the bandwidth is reserved, and that slots reserved do not immediately occur after the handshake temporally. A further difference is the way bandwidth is spatially reused. Here, the bandwidth is more aggressively reused, and no bandwidth is reserved for the reverse link for an Acknowledgment (ACK) via the three-way handshake. The slot states are thus used by nodes in the WMN to keep track of the transmissions already scheduled in their neighbourhood, and thereby find out which slots are still free for scheduling further non contending transmissions. Correctly updating the slot state is not trivial. Details of our design for slot-state update can be found in Ref. [83].

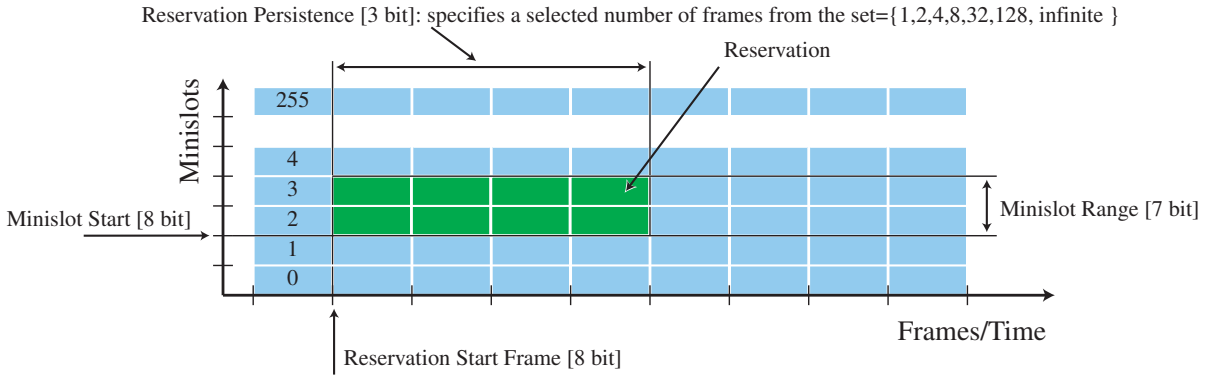


Figure 3.5: Visualization of Reservation Using Distributed Scheduling

Fig. 3.5 visualizes the concept of bandwidth reservation in the MeSH mode. Each grant (grant confirm) identifies a rectangular (see Fig. 3.5) area of slots which are reserved. A given contiguous range of slots can be granted for a contiguous set of frames. Thus, the reserved slots are identified by specifying the start frame and frame persistence, and minislot start and the number of minislots reserved. Frame persistence identifies the number of frames for which the specified minislots are being granted. Frame persistences permitted in the standard allow the values (number of frames) from the set {1, 2, 4, 8, 32, 128, ∞}.

Thus, a reservation specifies a range of slots (ΔMS) and a number of frames (persistence) for which the range of slots is reserved (denoted here as $Per_{\Delta F}$. Where $\Delta F \in$ {1, 2, 4, 8, 32, 128, ∞}. Nodes in the WMN need to explicitly free slots reserved with persistence Per_{∞}. This is done by using a special persistence Per_0 to indicate cancellation of slots previously reserved using Per_{∞}. Shorter duration reservations (i.e. with persistence smaller than Per_{∞}) cannot be cancelled. Thus, slots once allocated with such persistences remain blocked till the reservations are no longer valid (i.e. the frames for which the slots were reserved have passed).

We now look into the details of the messages used by the nodes for the three-way handshake. Nodes use transmission opportunities in the schedule control won by themselves (using a distributed MeSH Election, explained later) to transmit messages for the three-way handshake. The message which is transmitted is called MSH-DSCH message shown in Fig. 3.6. The MSH-DSCH message is the payload of the MAC-Protocol Data Unit (PDU) which is actually transmitted by

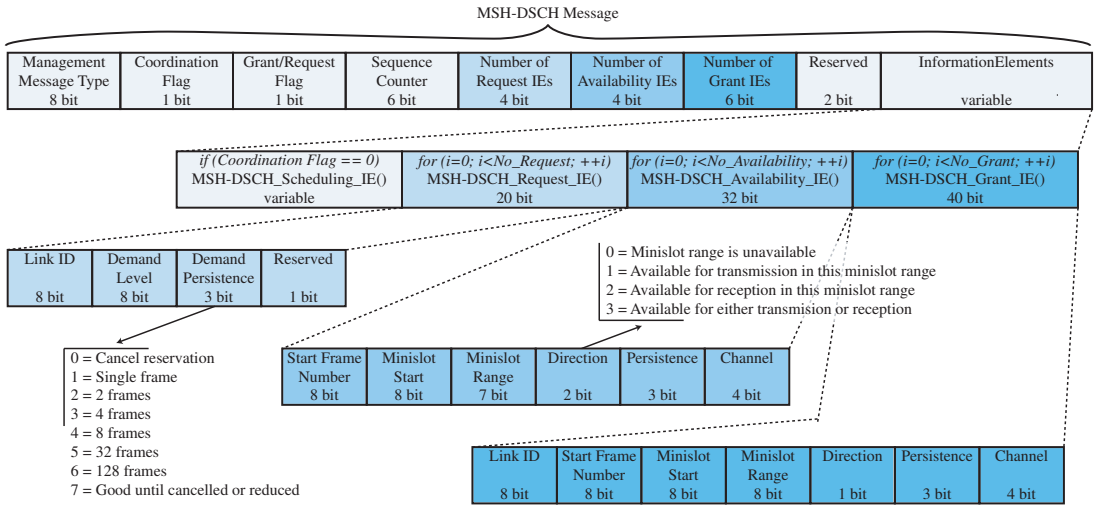


Figure 3.6: MSH-DSCH Message Structure

the node. Fig. 3.6 shows all the details of the MSH-DSCH message. Here we will discuss only the most important details, for further details the readers are referred to [42, 83].

As seen from the figure, a single MSH-DSCH message may contain a set of requests (Request_IEs), a set of grants and grant confirmations (using Grant_IEs), and a set of availabilities (Availability_IEs). A Request_IE specifies the bandwidth needed to be reserved, i.e. it specifies the number of slots to be reserved, and the number of frames for which the node would like to reserve these slots. The Availability_IEs specify the slots which are suitable for transmission at the node sending the MSH-DSCH message. These can be used by the node which issues a grant to select suitable slots for granting as we discussed previously. The Grant_IEs are used for both grants as well as grant confirmations. These specify the actual slots which are being reserved. The standard does not specify how the individual Information Elements (IEs) are packed into a MSH-DSCH message. We have designed custom and sophisticated solutions for the same, details of one of which can be found in Ref. [81].

Nodes transmit the MSH-DSCH message using transmission opportunities which they have won via the distributed MeSH election process. We will provide a simple explanation of the working of the MeSH election process using Fig. 3.7. *Current Xmt Time* identifies the transmission opportunity a node has currently won via the election and the opportunity in which it will transmit a MSH-DSCH message. At this time a node uses the MeSH election to compute the time when it will next transmit a MSH-DSCH message. Note that the MeSH election ensures that the MSH-DSCH messages transmitted by the nodes in the control subframe can be received without collision via all the neighbours of the node transmitting the message. Each node advertises in the MSH-DSCH message an interval which contains the actual next transmission opportunity won by the node. This enables its neighbours to know that during this interval the node will be transmitting a MSH-DSCH message. The neighbours however, do not know the exact transmission opportunity in which the node actually transmits the MSH-DSCH message.

Each node thus has information about the *Next Xmt Time* intervals of its two-hop neighbours. The nodes also have information about the *Advertised Xmt Holdoff* for their two-hop neighbours. This specifies the number of transmission opportunities for which a node will not try to access the control subframe after transmitting once in the control subframe in an opportunity it had won via the election earlier. As seen from Fig. 3.7 different nodes may have different transmission holdoff intervals. Nodes can compute for each neighbour the earliest subsequent transmission time by simply adding the advertised transmission holdoff to the nodes advertised next transmission interval.

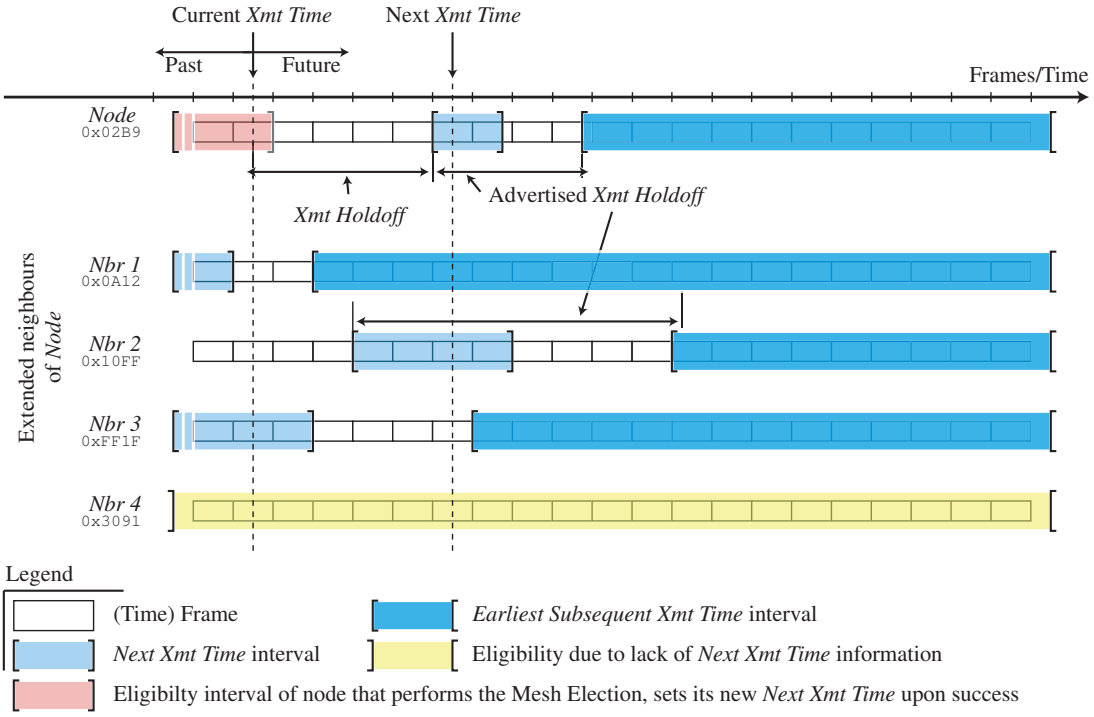


Figure 3.7: MeSH Election to Access Control Subframe

Consider now the sample scenario in Fig. 3.7. The node with identifier 0x02B9 is at its *Current Xmt Time*. This node has four nodes in its two-hop neighbourhood as shown in the figure. Before transmitting the MSH-DSCH message the node needs to use the MeSH election process to find out its next transmission time, and also advertise the same along with its new transmission holdoff in the MSH-DSCH message it will transmit. Node 0x02B9 will add its holdoff time to the current transmit time to compute the earliest opportunity for which it may compete using the election. For this transmission opportunity it will then collect the set of nodes who are also competing for the transmission opportunity. A neighbour is considered to be competing for a given transmission opportunity if the opportunity lies within its *Next Xmt Interval* or if the opportunity lies after the earliest subsequent transmission time of the node. Nodes for which no valid information is available are assumed to be competing for any chosen transmission opportunity. Thus, in our example, node 0x02B9 will find that for the chosen transmission opportunity the neighbouring nodes *Nbr1*, *Nbr2*, and *Nbr4* are competing. It will then collect the node identifiers of all these competing node and input these along with its own node identifier and the number of the opportunity for which the nodes are competing to the MeSH election algorithm. The return value from the MeSH election process will be either a win or a lose for the given transmission opportunity for the node running the election. If the node wins the election, it knows its next transmission opportunity, and then computes an interval which contains this opportunity, and advertises this interval with a new value for the transmission holdoff in the MSH-DSCH which it transmits in the *Current Xmt Time*. If the node happens to lose the transmission opportunity, it continues the above process with the next transmission opportunity and so on till it manages to win one transmission opportunity. The MeSH election is nothing but a pseudorandom mixer and mixes the node identifiers and the transmission opportunity identifier randomly to uniquely let one node in a two-hop neighbourhood win any given transmission opportunity. Details of the pseudocode for the MeSH election can be found in [42, 83].

We thus can see that the three-way handshake duration depends on a lot of factors. The values for the transmission holdoff, and also the number of competing nodes in the two-hop neighbourhood of a node play a vital role in deciding the duration of a three-way handshake. Ref. [14] provides a simplified analytical model for the three-way handshake delay and the MeSH election process. Thus, once a node decides that it needs bandwidth for transmission, it will take at least a delay equal to the three-way handshake duration till it is able to transmit data using the slots reserved via the handshake.

In this chapter, we have seen the scheduling mechanisms in the MeSH mode and distributed scheduling in some detail. These details are vital as we use the MeSH mode to demonstrate the proof-of-concept for our research. In the next parts of this thesis we will look at the issues involved in efficiently deploying network coding solutions in the MeSH mode.

Part II

Local Mechanisms for Efficient Deployment of Network Coding

4 Design Issues and Principles for Deployment of Network Coding

As discussed earlier, in order to deploy network coding efficiently in reservation based WMNs we will be designing distributed solutions for network coding, which work by using locally available data and that which is available from the direct neighbourhood of a node. However, for an optimal deployment of network coding a global view of the wireless mesh network is also needed. In this part of the thesis we will be looking at the former (i.e. distributed local network coding solutions) in detail. The latter (WMN wide global network coding optimization), will be covered in Part III of this thesis.

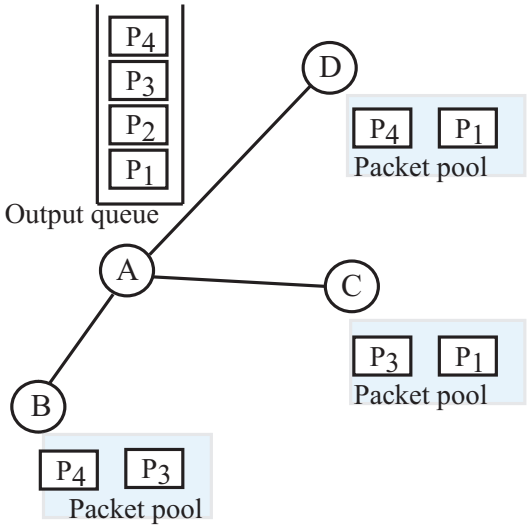
In this chapter we will look at the issues involved when designing network coding solutions for reservation based WMNs such as those based on the IEEE 802.16 standard in some depth. Here, we will identify the pitfalls in deploying conventional practical network coding solutions in such WMNs. Most of the conventional state-of-the-art practical network coding solutions have been designed and tested keeping in mind IEEE 802.11 based WMNs. Here, we will highlight why this leads to solutions for network coding which are not practically feasible for the reservation based IEEE 802.16 MeSH mode. In Sec. 4.1 we first guide the readers to the pitfalls via a simple thought experiment. This at the same time provides the readers with insights into the critical differences between IEEE 802.11 based WMNs and IEEE 802.16 based WMNs, which makes it necessary to look at network coding in such networks from a different perspective. Then, in Sec. 4.1.1 we present an analytical model for the bandwidth reservation procedure in the IEEE 802.16 MeSH mode, with a special emphasis on highlighting the critical points for designing network coding solutions for the MeSH mode. Finally, based on the prior discussion, and an analysis of our model, we derive design principles for practical network coding solutions in reservation based WMNs such as the MeSH mode (see Sec. 4.2). These design principles will be the guiding and driving principles for the distributed network coding solutions we present in Chap. 5.

4.1 Design Considerations for Network Coding in the MeSH Mode

Most of the conventional state-of-the-art practical deployment solutions for network coding in wireless mesh networks make use of the inherent broadcast nature of the wireless medium (e.g. see references [49, 48, 101, 28]). The seminal work of Katti et al. [49] for example makes use of opportunistic listening to enable opportunistic coding to efficiently enable network coding in IEEE 802.11 based WMNs. They propose the use of a simple XOR coding to encode and mix information contained in packets arriving at a node before transmitting the coded packets. As the design principles presented there form the basis for most of the practical network coding solutions in conventional WMNs, we will use this work to exemplify the issues and pitfalls involved when conventional network coding solutions are deployed in reservation based WMNs. Hence, we will look at the principles of opportunistic listening and opportunistic coding in depth to see the issues which would arise if such mechanisms are to be employed in the reservation based MeSH mode.

Consider the WMN topology shown in Fig. 4.1. The figure shows the packets currently in the output queue of node A. Thus the packets $P_1 \ldots P_4$ are currently waiting in the output queue

of node A for transmission. The table in Fig. 4.1 shows the routing decisions to be made by node A for the packets in its output queue based on the routing table entries at node A. For example, based on its routing table node A should forward the packet P_1 to node B, and the packet P_2 should be forwarded by node A to node D and so on. Now, also consider that node A has knowledge about the packets each of it's neighbours have saved locally in their individual packet pools. Each node is supposed to maintain a local packet pool. This packet pool is used by the nodes in the WMN to temporarily save and store packets the node itself has transmitted in the past, received in the past addressed to itself, or packets which have been overheard by the node from its neighbours. Such packets locally available in the packet pools are then used by the nodes for either coding or decoding operations when using network coding in the WMN. Thus, we assume that node A knows for the given example, that node D has packets P_4, and P_1 in its packet pool, and node C has stored packets P_3 and P_1 in its packet pool, and that node B has stored packets P_4 and P_3 in its packet pool.



Packet to be transmitted at node A	Next hop for packet
P_1	B
P_2	D
P_3	D
P_4	C

Figure 4.1: Example for Opportunistic Listening and Coding

The node A obtains information about the packets stored by its neighbours in their packet pools via reception reports which are periodically transmitted by the nodes, or via intelligently guessing which packets the neighbours may have in their pools based on the packets the node A has overheard on the wireless medium in its neighbourhood (see Ref. [49] for details).

Based on the available information, node A has different choices as to which packets to code together via XOR before transmission. In principle it is possible for node A to code by XORing together any two or more packets currently pending for transmission in its output queue to form a coded packet which it can transmit. For example, node A can decide to code packets P_3 and P_1 and transmit $P_3 \oplus P_1$. Transmitting this combination will enable node D to correctly decode and receive packet P_3 and will enable node B to correctly decode and receive packet P_1. Thus, in a single transmission two packets pending in the output queue at node A will reach their next hops, this is in contrast to the two separate transmissions which would be needed in case node A would have transmitted these packets uncoded. To decode and receive a packet a node will look at the information in the coded packet which tells which packets are coded together, and then XOR the corresponding packets from its own packet pool with the received encoded packet to decode and obtain the packet it is missing.

Another possibility for node A is to encode packets P_1, P_3, and P_4 together and transmit the encoded packet $P_4 \oplus P_3 \oplus P_1$. Transmitting this encoded packet is better than transmitting the encoding we discussed earlier. This is because in this case three packets from the output queue

of node A reach their next hops instead of only two packets using the earlier coding. In this case, packet P_3 can be decoded by node D, packet P_4 can be decoded by node C, and packet P_1 can be decoded by node B. Thus, transmitting this combination is a better choice in comparison to the previous encoded packet as more information can be exchanged using a single transmission. This demonstrates the use of opportunistic listening and opportunistic coding. Nodes in the WMN opportunistically listen to packets being transmitted in their neighbourhood, and given the opportunity, by observing the currently available packets in their output queues, and the information of packets available with their neighbours for decoding choose a set of packets to XOR together to form a coded packet to transmit. In the above scenario, it would also have been possible for node A to transmit the combination $P_3 \oplus P_2$. However, this combination is meaningless as not even a single packet reaches its intended next hop via this transmission. Thus, at each stage the node coding packets needs to make an informed and intelligent choice about the packets it should code and transmit. And this decision about coding can be made by a node only after it has the packets waiting for transmission in its output queue, and only after it has up to date information about the information available with its neighbours. Thus, for each coded transmission, a node may choose packets belonging to different sets of streams for coding.

In Ref. [49], the authors show that the above simple approach is practically applicable in WMNs using the IEEE 802.11 standard and can demonstrate significant bandwidth savings due to network coding. The above approach is, however, extremely inefficient in reservation based WMNs such as the MeSH mode. Let us assume in our example that node A decides to code the locally available packets as: $P_4 \oplus P_3 \oplus P_1$. This combination should then be transmitted such that it is received at all the neighbours of A (nodes B, C, and D). In the MeSH mode this means that node A needs to have bandwidth reserved a priori for transmission to the set of neighbours in question before it can transmit the coded packet. The MeSH mode permits reservation of bandwidth only for transmission on individual links in the WMN according to the definitions in the standard. Thus, without amendments to the standard, it is not possible to reserve bandwidth for transmission to multiple nodes simultaneously. This implies that the node wanting to transmit the coded data to multiple neighbours simultaneously needs to perform a three-way handshake with each of the neighbouring nodes. Even if the standard were to provide mechanisms for an extended handshake which permits reservation of bandwidth for broadcast (transmission to all neighbouring nodes), or multicast (transmission to a subset of a node's neighbours), getting the required amount of bandwidth in time for such broadcast or multicast data transmissions is a non trivial task. In Sec. 4.1.1 we will analytically model the bandwidth reservation handshake in order to enable the readers to gain a better insight into the issues which need to be considered when designing practical network coding solutions for reservation based WMNs.

4.1.1 Analytical Model for the Handshake Based Bandwidth Reservation and its Implications for Network Coding Solutions

We will next present an analytical model for the bandwidth reservation process in the IEEE 802.16 MeSH mode in order to gain better insights into the critical challenges for practical network coding solutions in reservation based WMNs. Let us consider the parameters listed in Table 4.1 which are used in our analytical model. We will consider only distributed scheduling to be used for bandwidth reservation in the MeSH mode as centralized scheduling is only of limited use in WMNs (see Chap. 3 for details).

Let us say that we have a total of M slots available for distributed scheduling in the data subframe. M_{Tx}^{t} then denotes the number of slots which have status suitable for scheduling data

Table 4.1: List of Parameters for Analytical Model for the Distributed Scheduling

Parameter	Interpretation of Parameter
M	Num. of slots for distributed scheduling in a frame
M_{Tx}^{t}	Num. of slots suitable for scheduling transmission at the sender t (i.e. slots with status *av* or *tav*)
M_{Rcv}^{k}	Num. of slots suitable for reception at receiver k (i.e. slots with status *av* or *rav*)
t	The node wanting to transmit the (coded) data, it is the node which initiates the handshake(s)
K	Num. of receivers to which the transmission is to be scheduled
k	$1 \ldots K$, index for intended receivers respectively
d	Num. of slots to be reserved (demand)
C_T, C_{R_k}	$\binom{M}{M_{Tx}^{t}}$, $\binom{M}{M_{Rcv}^{k}}$ respectively

transmissions at the node t. These are slots with slot state either *tav* or *av*. Similarly, M_{Rcv}^{k} denotes the number of slots which have status suitable for scheduling reception of data at the neighbour of node t indexed by k. Now, we see from the table that the transmitter t wants to transmit a coded packet to K neighbours, which are indexed and identified by the variable $k \in 1 \ldots K$. The transmission of the coded packet requires d slots. We assume that the parameters M_{Tx}^{t} and M_{Rcv}^{k} hold for a given frame, and for each frame we can have different values for each of these parameters.

We now look at the pitfalls in implementing COPE-like [49] practical network coding solutions in the WMN using the MeSH mode. A core principle of COPE's packet coding algorithm is to not delay the transmission of packets just for the sake of enabling coding of packets. This is very important, especially in the case of reservation based WMNs where bandwidth reservations are used to support hard QoS guarantees. Using COPE, a node will look at its output queue and choose a set of packets which it can code together based on the information it has about the packets available at its neighbours. COPE can code and transmit packets as soon as a set of matching codable packets are available at the transmitting node.

This is not the case for the MeSH mode due to its reservation based nature. Let us go back to the example presented in Fig. 4.1 to understand this. Assume that node A decides to code the packets P_3 and P_1 to give the Coded Packet $CP{=}P_3 \oplus P_1$. Using COPE this coded packet *CP* would be then broadcast by the node A. However, this packet is meaningful only for a subset of the neighbours of node A, namely, nodes B and node D. In our model we would then have $K{=}2$. Let d be the number of slots required for transmitting the coded packet *CP*. In this example, the node t then corresponds to node A, as this is the node which wants to schedule the multicast transmission of the coded packet *CP* to a subset of its neighbours. Now, as all the data transmissions in the MeSH mode are to be scheduled in a contention free manner, bandwidth needs to be reserved for the multicast transmission by node A before it can transmit packet *CP*. This means that the node A has to wait until it has managed to reserve d slots in a given frame for the multicast transmission of the coded packet *CP* to its neighbours B and D. Note that here we face the first shortcoming in the MeSH mode. The MeSH mode does not natively support a handshake procedure for reserving bandwidth for multicast. Also as seen in Chap.

3, for the three-way handshake for unicast bandwidth reservation, nodes use the slots in the control subframe, which in turn are accessed via the distributed mesh election procedure. This, in turn, further adds to the latency of bandwidth reservation. This delay will only get worse if bandwidth for multicast is reserved via multiple modified three-way handshakes, one each with every recipient for the multicast transmission. The MeSH mode does not natively support multicast bandwidth reservation, and this is not without reason.

For the following discussion let us assume that we use enhanced handshake procedures that allow us to reserve multicast bandwidth. Let us also assume as in the above example that the node t wants to transmit the coded packet CP to K neighbours, and needs to reserve d slots for the transmission. As shown in Tab. 4.1, we assume that in the frame where the transmitter t wants to reserve the d slots, it has M_{Tx}^{t} slots with slot state suitable for transmission. Also let us assume that in the same frame the intended receiver identified by the index $k \in 1 \ldots K$, has M_{Rcv}^{k} slots with status suitable for scheduling reception of data. For the multicast reservation to be possible at all in the given frame, we require $|M_{Tx}^{t} \cap M_{Rcv}^{k}| \geq d$, $\forall k$. If this condition is not met even for a single receiver (the nodes to receive the multicast transmission of the coded packet) or at the transmitter then it is not possible to reserve the needed number of slots in that frame, and one needs to then try to reserve the slots in some other frame (possibly farther off in the future). However, even if the latter condition is satisfied for each individual receiver, it does not guarantee that bandwidth reservation is possible. In order for bandwidth reservation to be possible each of the receivers and the transmitter should at least have exactly the same d slots available in the suitable status. Thus the condition $|M_{Tx}^{t} \cap M_{Rcv}^{1} \cap M_{Rcv}^{2} \cdots \cap M_{Rcv}^{K}| \geq d$ should be satisfied for the multicast bandwidth reservation to be possible at all. For the model parameters given in Tab. 4.1, for a given frame, using counting theory we can derive the probability of successfully being able to find d slots suitable at both the transmitter and all the receivers as given in Eq. (4.1).

$$P_{succ} = \frac{C_T \prod_{k} \sum_{j=d}^{\min(M_{Tx}^{t}, M_{Rcv}^{k})} \binom{M_{Tx}^{t}}{j} \binom{M - M_{Tx}^{t}}{M_{Rcv}^{k} - j}}{C_T \left(\prod_{k} C_{R_k}\right)} \tag{4.1}$$

Fig. 4.2 and Fig. 4.3 show plots of P_{succ} for different values of the involved parameters in Tab. 4.1 as given by Eq. (4.1).

The total number of slots (M) per frame is assumed to be 100. The x-axis shows the number of slots suitable for transmission (i.e. status *av* or *tav*) on the transmitter's side. The y-axis shows the number of slots suitable for reception at the receiver(s) (i.e. slots with status *av* or *rav*). For ease of plotting we have plotted only the case where all the receiving neighbours are assumed to have the same number of slots available for reception. Comparing Fig. 4.2 (a) and Fig. 4.2 (b), (c), and (d) we notice that a higher number of slots need to be available for transmission at the sender, and a higher number of slots need to be available at the receiver(s) with an increase in the number of slots to be reserved (d), to successfully reserve the required slots in a given frame with a high probability. In short, the probability of successfully reserving d common slots for transmission to a fixed number of receivers (K) in a given frame decreases with increasing d, given that the number of receivers, the number of available slots at the transmitter and the receiver(s) remain unchanged. In all the above figures we have K=1, which basically means that we have plotted the case for successfully reserving d slots in a given frame for a unicast transmission. Even in this case, a significant number of the total available slots have to be in the appropriate slot states at both the transmitter and intended receiver in order that the handshake be successful.

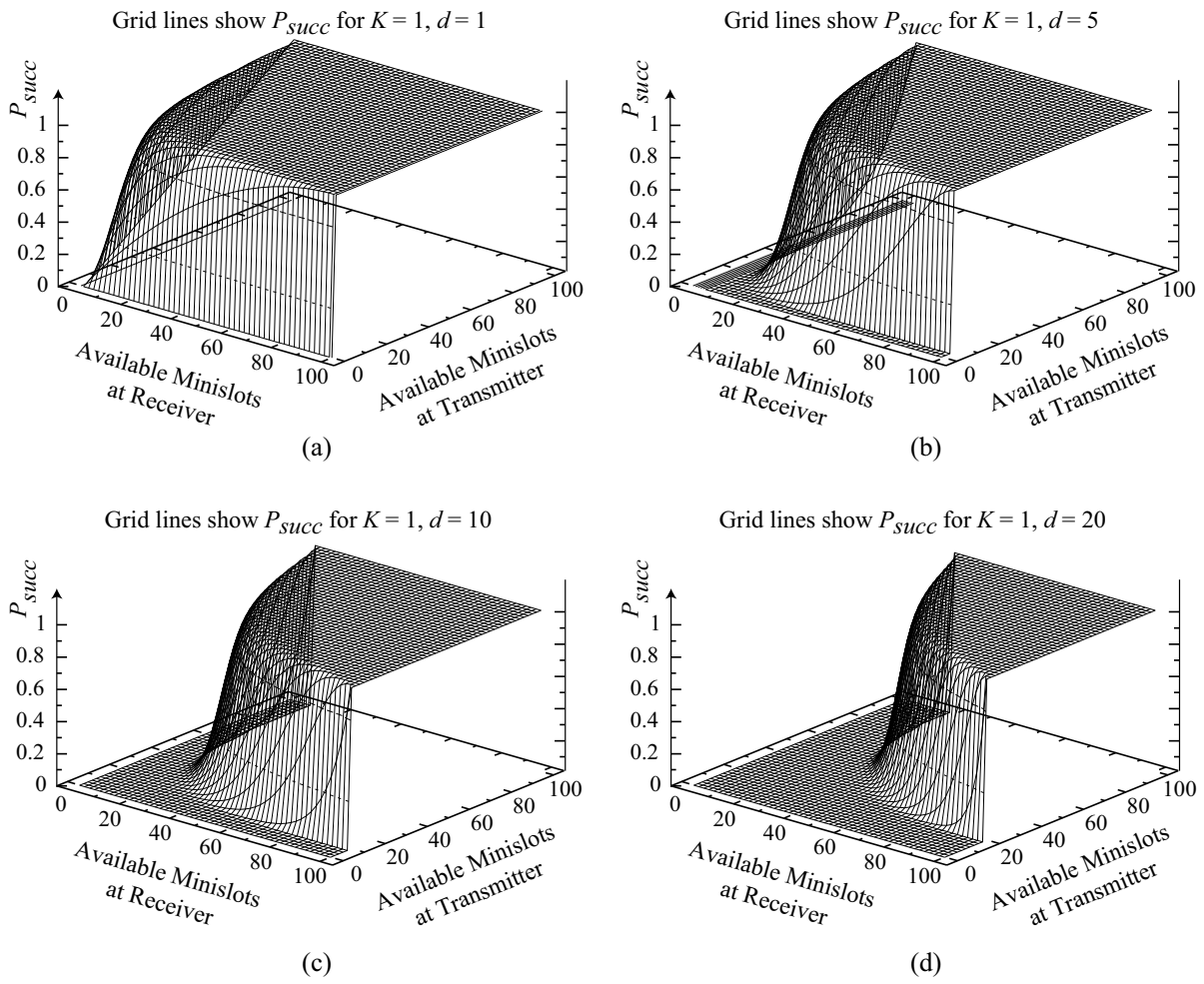


Figure 4.2: Plots Showing the Probability P_{succ} of Successfully Reserving d Slots in a Given Frame for Simultaneous Reception at K Neighbouring Nodes: (a) $K = 1$, $d = 1$; (b) $K = 1$, $d = 5$; (c) $K = 1$, $d = 10$; (d) $K = 1$, $d = 20$

We now also have a look at the effect of an increase in the number of neighbours to whom the bandwidth is to be reserved in addition to the effect of d on P_{succ}. Consider Fig. 4.3. It shows the contour plots showing contour lines beyond which the value for P_{succ} exceeds the given threshold. This means that on the graph above the contour line we have a value of P_{succ} which exceeds the threshold mentioned on the graph. Here too, for ease of plotting and interpretation of the plots we have assumed that all the receivers have the same number of slots in a status suitable for scheduling reception of data. The transmitter is the node which wants to transmit the coded packet *CP* to a subset of its neighbouring nodes.

Comparing Figs. 4.3 (a) and (b) or Figs. 4.3 (c) and (d) we see that for the same demand d, the number of suitable slots needed at the transmitter and receiver(s) for successfully reserving (with a certain probability of success) the required number of slots increases with the number of receivers (K) involved in the handshake. Analysis of the figures and Eq. (4.1) shows that P_{succ} decreases drastically even if either the transmitter or one of the receivers has only a small number of slots suitable for the intended communication in the given frame. Further, with increasing d, a large number of slots needs to be free for the intended communication, to be able to successfully negotiate and reserve slots with a high probability. For the same demand d and the given number of free slots at both the transmitter and receivers, P_{succ} decreases with an

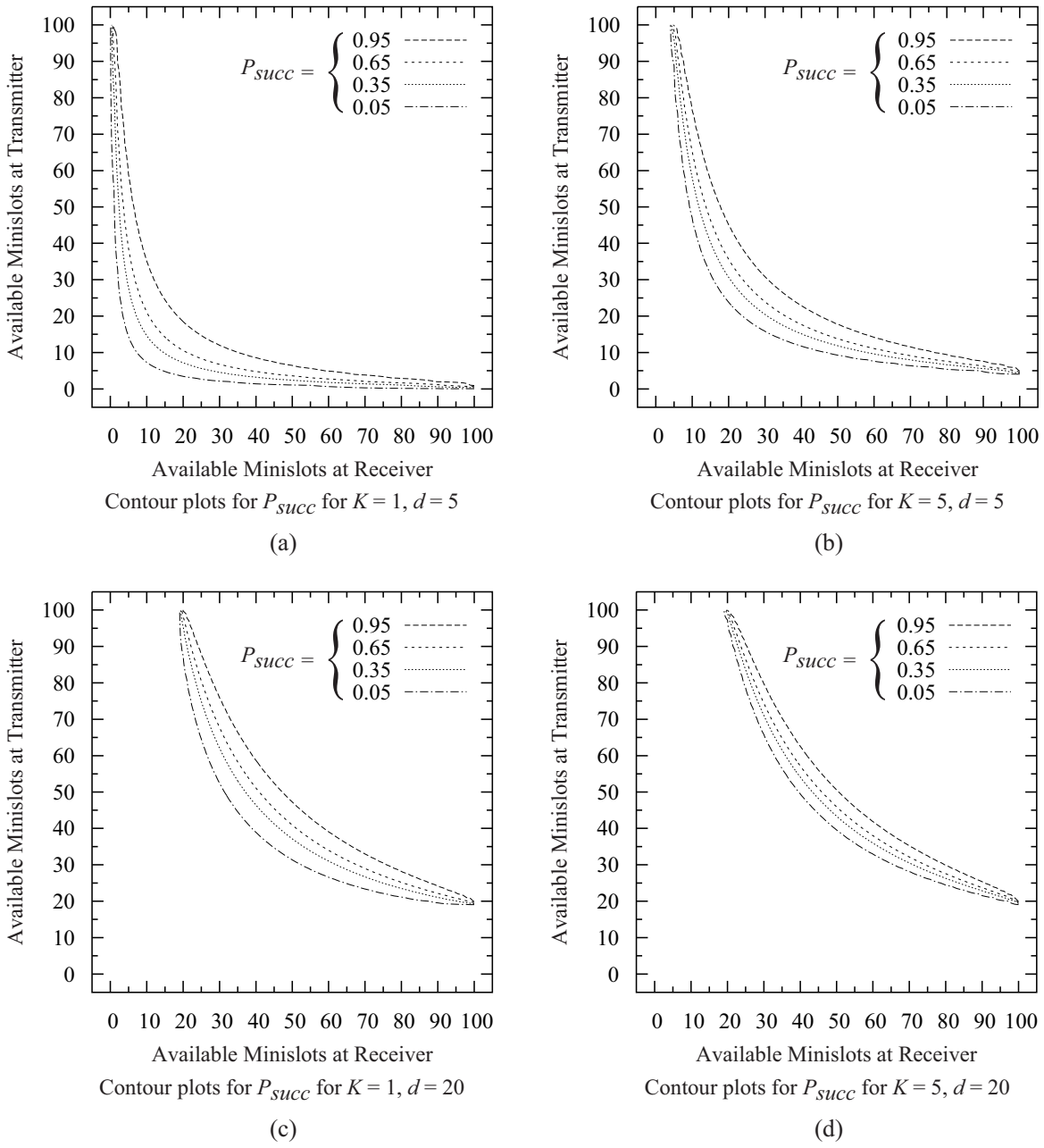


Figure 4.3: Contour Plots Showing the Probability of Successfully Reserving d Slots in a Given Frame for Simultaneous Reception at K Neighbouring Nodes: (a) $K = 1$, $d = 5$; (b) $K = 5$, $d = 5$; (c) $K = 1$, $d = 20$; (d) $K = 5$, $d = 20$.

increase in the number of intended receivers K (see Fig. 4.3). In practice, not all receivers share the same number of available slots; a single receiver having a low number of slots suitable for reception results in P_{succ} to be very low. We can conclude that on-demand reservation of slots for network coding transmissions cannot be achieved with high success, which means that—unlike COPE—we need to set up the reservation for the multicast network coding transmission prior to the arrival of a set of packets which can be coded. Additional results showing the trends for P_{succ} can be found in Appendix C in Fig. C.1.

An important aspect of the IEEE 802.16 MeSH mode is the way the reservations are carried out. Nodes perform the three-way handshake to reserve bandwidth for links to individual neigh-

bours; distributed scheduling messages (MSH-DSCH) containing *Request*, *Grant-confirmation*, and *Grant* are exchanged to reserve a set of slots for the required transmission. In the above analysis we have computed the value for P_{succ} with some simplification. Firstly we have considered that the entire range of available slots at the transmitter and all the receivers is available for finding a suitable intersecting match of slots suitable for all partners involved in the handshake. However, due to message size restrictions, with the bandwidth request in a MSH-DSCH message, the transmitter can only advertise a subset of the slots suitable for transmission to the receivers. This effectively reduces the value of P_{succ} by reducing the number of slots available at the transmitter for negotiating the reservation. Secondly we have assumed that if a common set of d suitable slots can be found at the transmitter and each of the receivers then the handshake is successful. However, a more important problem with multicast reservation is that each node maintains its own independent state for all the minislots. Thus, the individual receivers for the multicast transmission do not possess a common view about which slots are suitable at the other receiver(s) and may, hence, issue grants for different slot ranges (although it may have been possible to negotiate a common set of slots). Such disjoint grants require multiple transmissions (one to each neighbouring node in the worst case), thereby defeating the goal that coded transmissions should be simultaneously received by multiple neighbours.

Another critical aspect that needs to be addressed by network coding solutions designed for 802.16's MeSH mode is the three-way handshake overhead. This aspect should be also considered for any reservation based WMN, and when designing network coding solutions for reservation based WMNs. Any reservation mechanism brings with itself a certain overhead in terms of bandwidth requirements, as well as latency which needs to be considered.

In the MeSH mode each node may transmit the control messages (MSH-DSCH) for the three-way handshake only in transmission opportunities belonging to the control subframe, which have been won by the node using the mesh election algorithm specified by the standard. Ref. [14] provides an analytical model for mesh election and analyzes the three-way handshake delay. Let the mean three-way handshake duration between transmitter t and receiver k be H_k^t. The standard's scheduling constraints require that slots granted by the receiver may be used for transmission only after the three-way handshake is complete. Hence, it is only meaningful to grant slots in frames occurring after the completion of the three-way handshake. For a multicast handshake, as required for network coding, it implies that the nodes should start searching for the required d slots in frames after a duration $T_H^K = \max_k(H_k^t)$. Let P_{succ_i} be the probability of successfully being able to reserve the required slots in frame i for the intended communication (i.e. given a set of transmitter, receivers, and d required slots and M total slots in the frame).

The mean number of frames that need to be considered starting from a given start frame to reach the first frame in which the demand can be satisfied is given by Eq. (4.2).

$$F_{mean} = \sum_{n=1}^{\infty} n\, P_{succ_{(sf+n)}} \prod_{j=1}^{n-1} \left(1 - P_{succ_{(sf+j)}}\right) \tag{4.2}$$

Here, *sf* is the number of the frame after completion of the multicast handshake. This is an optimistic estimation as we assume that all the nodes start attempting to find the required slots in the frame right after the last handshake with one of the receiver's is over. In case nodes start further off in the future with their search for suitable slots then *sf* will be higher. Assuming that the duration of a frame is F_D, the mean waiting time before the reserved frame for the multicast transmission starting from the start of the multicast bandwidth reservation handshake is given by Eq. (4.3).

$$T_{mean} = T_H^K + F_{mean}\, F_D \tag{4.3}$$

Note, as discussed earlier, the above is an optimistic and simplified estimate. In reality, with incomplete knowledge about the slots available at the other nodes involved in the multicast handshake, and without knowledge about the current slots which have been granted by some nodes as the handshake is running, the probability of the handshake failing increases considerably. Hence, from the above analysis we can conclude that just in time opportunistic reservation of slots for network coding, and an opportunistic network coding scheme based on opportunistic listening like in COPE is infeasible for reservation based WMNs. Therefore, when designing network coding schemes for reservation based WMNs a new approach is necessary and alternative design principles for network coding solutions need to be considered. Sec. 4.2 presents design principles for network coding solutions for reservation based WMNs which build up on the insights obtained by the analysis presented so far.

4.2 Design Principles for Network Coding in Reservation Based WMNs

In the previous section, we analytically modelled the handshake procedure for bandwidth reservation with a view on highlighting the pitfalls involved when contemporary opportunistic network coding schemes are deployed in reservation based WMNs. From our analysis we can obtain the following set of guiding design principles for designing network coding solutions for reservation based WMNs.

- ***Principle 1*** (DP_1): On demand reservation of multicast slots for Wireless Network Coding (WNC), i.e. reserving slots after a set of packets for coding is present, is not feasible without prohibitive overhead; hence, reservation of multicast slots should ideally be performed a priori.

Principle DP_1 is a core principle which needs to be kept in mind when designing network coding solutions for reservation based WMNs. This is of special importance if exorbitant delay due to network coding is not acceptable. This is usually the aim of all network coding solutions, and is of vital importance for multimedia traffic which cannot tolerate high delays and needs to be routed to its destination within specified delay bounds. Increasing delays due to network coding in order to save bandwidth basically defeats the purpose of using a reservation based WMN in order to ensure low delays due to guaranteed bandwidth availability. Therefore, DP_1 proposes that one should avoid using network coding solutions in reservation based WMNs which attempt to reserve multicast bandwidth for network coding once suitable packets are available at the node which codes the packets and suitable packets are also available at the neighbouring nodes for decoding the coded transmission. This in turn means that we can no longer make network coding decisions by looking at individual packets but need to radically move away from this approach. DP_1 suggests that such multicast bandwidth reservations for network coding should be ideally established prior to the coding of the packets itself. However, persistently blocking and reserving bandwidth for all possible multicast combinations for network coding, at all nodes in the WMN, is not a suitable option. Thus, means need to be devised to intelligently detect which multicast network coding transmissions offer the maximum promise of bandwidth savings, and that over a longer duration such that the overhead of bandwidth reservation can be amortized by the savings due to network coding. Once such promising network coding constellations are detected then one can reserve bandwidth for the necessary multicasts over a large number of frames. We have in this thesis designed and presented efficient solutions for the above (for details see Chap. 5).

- ***Principle 2*** (DP_2): The higher the number of neighbours in the multicast reception set, the more difficult it is to get an agreement on a common set of slots for reception, especially in presence of background traffic and different number of available slots at the involved

parties. The success probability of such a reservation in a given frame further diminishes with an increase in the demanded slots. Hence, the size of the receiver set should be kept as small as possible.

Design principle DP_2 follows directly from our analysis in Sec. 4.1.1. From Eq. (4.1) and Figs. 4.2, 4.3 we have seen that the probability of successfully reserving bandwidth for a multicast transmission for a given frame reduces drastically with an increase in the number of intended receivers for the multicast. From DP_1 we see that it is beneficial to reserve bandwidth for multicasts not for just a single frame but for a larger number of frames in order to offset the overhead involved with the bandwidth reservation mechanisms. It is even more difficult to successfully reserve bandwidth for a multicast transmission for a contiguous set of frames as compared to reserving the bandwidth for a single frame. This is because the same common set of d slots needs to be available for the multicast reservation in all the frames over which the bandwidth reservation is to be valid. Hence, in practice, when designing practical network coding solutions for reservation based WMNs one should avoid solutions which would require a larger number of nodes to receive the coded transmissions simultaneously. This is, however, not a limitation for network coding solutions for WMNs for the scenario which we are tackling in this thesis, namely that of network coding solutions for multiple unicast streams in the WMNs.

In the empirical study presented in Ref. [49] it is found that around 50% of network coding operations involve only two symbols (i.e. two packets coded together, which implies that two nodes need to receive this coded packet). In similar settings, however, with an aim to minimize energy requirements the authors found out in Ref. [19] that less than 1% of coding operations involved combination of more than three symbols (i.e. three or more packets combined togehter). Similarly, following the above reasoning the authors in [52] limit their Pairwise Intersession Network Coding (PINC) approach to coding between only pairs of coexisting traffic sessions. Thus, the guideline provided by DP_2, imposes for all practical purposes, only a minor limitation on the benefits which can be obtained via network coding under normal operating conditions. Only under special conditions is it possible to code over more than two symbols. And in reservation based WMNs, network coding, where a large number of nodes are in the multicast receiver set brings more pitfalls than gains which can be obtained via network coding (see the discussion in Sec. 4.1).

- ***Principle 3*** (DP_3): The three-way handshake delay combined with the overhead of reserving the required multicast slots means that the number of such three-way handshakes required should be kept to a minimum. If possible, the handshake should optimize the probability of getting a successful multicast reservation.

Design principle DP_3 aims to keep the reservation overhead required for network coding to a minimum. Ideally, reservations should be done for network coding multicasts for a large number of frames. Thereby it is possible to amortize the overhead of the handshake (both delay as well as bandwidth requirement) via the benefits obtained by coding multiple packets and multiple coded transmissions. Additionally, in most cases in reservation based WMNs, as is the case for the MeSH mode, the handshake procedures for bandwidth reservation are not optimized for network coding. Hence, in DP_3 we suggest that the handshake procedures should be modified, extended and adapted keeping in mind the special needs of bandwidth reservation for network coding. In Chap. 5 we present modifications to the three-way handshake used for bandwidth reservation in the MeSH mode to optimize it for network coding. The modifications to the reservation procedures should strive to not only speed up the multicast reservation but also make the handshake more robust and efficient by including mechanisms to enable nodes to arrive at an agreement on the common set of slots to be granted. This will not only reduce the delay for the handshake but also enhance the probability of a successful handshake for multicast bandwidth reservation. Another aspect which such modifications should consider is that they

should avoid wastage of bandwidth due to network coding (e.g. blocking nodes from scheduling transmission or reception of data needlessly).

In the next chapter we present our new paradigm for network coding which makes network coding decision at the level of streams (see definition in next chapter) of packets. Hereby, our solution departs from the packet-by-packet opportunistic network coding solutions and is suitable for reservation based WMNs needing scheduled access to the wireless medium. Our solution builds on the design principles and insights obtained from the discussion in this chapter which allows its efficient deployment in reservation based WMNs.

5 Distributed Mechanisms for Efficient Detection of Network Coding Opportunities and Deployment of Network Coding

In the previous chapter we saw some of the issues involved in reservation based WMNs which make a conventional, and opportunistic deployment of network coding in such WMNs difficult and quite expensive. Based on our discussion we derived a set of design principles which should be considered when drafting network coding solutions which cater to reservation based WMNs. In this chapter we will present our solution which we call "Stream-Oriented Network Coding (SONC)", which builds up on the insights obtained in the previous chapter and is designed to allow efficient and stable network coding in reservation based WMNs. To make the research more tangible we will develop SONC keeping the IEEE 802.16 MeSH mode as a concrete deployment scenario in focus. The concepts presented and developed are, however, generic and can be adapted and applied to other reservation based WMNs too.

We first present a bird's-eye-view overview of SONC with the use of simple examples showing the basics of SONC. We also introduce the readers to key definitions and notations we use throughout the development of SONC. This is followed by a detailed description of different phases of operation of SONC along with the details of the individual components involved in the operation of SONC.

5.1 SONC Core Principles and Terminology

In Sec. 4.2 we outlined the basic design principles which are of importance when designing network coding solutions for reservation based wireless mesh networks. Our solution SONC is designed keeping these insights in mind. We will motivate and explain the underlying concepts of SONC with the help of a simplified example. Based on our application scenario, and the arguments as discussed in Sec. 2.3, Sec. 4.1, and in Ref. [49] we will focus on an application of network coding for inter-session network coding, whereby we assume that the individual flows in the network are unicast flows, which, is a typical application scenario for the wireless mesh networks which we are targeting.

In this thesis we are interested in the simple XOR-type network coding, where the coded packets should be decoded at the immediate next-hop nodes. This is to avoid additional routing and network coding overhead in the WMN, especially in the application scenario with multiple unicast flows in the WMN, where each flow is dynamically changing. See Sec. 2.3 for additional justifications.

Let us say that the coded packet CP is given by the coding of the packets P_1, P_2, …, P_n. Using XOR coding CP is given by Eq. (5.1).

$$CP = P_1 \oplus P_2 \oplus \cdots \oplus P_n \tag{5.1}$$

Let us assume that as per the routing requirements the packet P_i $(1 \leq i \leq n)$ should be decoded at node n_i receiving the coded packet. To ensure that the node n_i is able to immediately decode

packet P_i we require that node n_i has already has the uncoded packets in the set {{ P_1, P_2, …, P_n }\{P_i}}. If a single packet from the above set is missing, it is not possible for node n_i to immediately decode P_i from the information it has, and the node must wait till further packets which are needed arrive before P_i can be decoded from the CP. When coding time-sensitive data, the relaying node coding the packets should avoid coding packets such that the desired packets are not immediately decodable at the next-hop. We will keep this constraint in mind when designing SONC.

Consider the network topology shown in Fig. 5.1. The wireless mesh network consists of seven nodes, with links between the nodes as shown in Fig. 5.1.

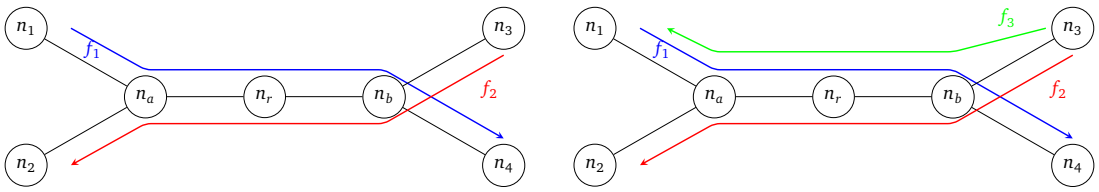


Figure 5.1: SONC Concept Visualization

Fig. 5.1(a) shows the WMN with two flows f_1 and f_2 existing in the WMN. We use the definition of a flow similar to that proposed in Ref. [90]. Thus, for the purpose of this thesis, when we talk of a flow we refer to a sequence of packets sent from a source to a particular destination (as we do not consider multicast destinations). Thus, as defined a flow can be composed of multiple end-to-end transport connections. We use the notation $f(s,d)$ when referring to a flow between source node s and destination node d. Thus, using our notation we have the flows $f_1(n_1, n_4)$ and $f_2(n_3, n_2)$ in the WMN in Fig. 5.1(a).

Let P_i^f denote the i^{th} packet belonging to flow f (let us assume we number packets sequentially starting from 1). Consider now Fig. 5.1(a) with the flows f_1 and f_2. Now, in this constellation, at node n_r, it is possible to use network coding to reduce the number of transmissions needed similar to the "Alice-Relay-Bob" example discussed in Sec. 2.3 and in Ref. [49]. From the view point of node n_r it is irrelevant which hops the packets for flow f_1 have prior to node n_a and it is also not relevant which next-hop the packets belonging to flow f_1 will be routed to after node n_b as far as network coding at the node n_r using XOR coding as in Ref. [49] is concerned*. Similarly, for network coding purposes at n_r it is also irrelevant which previous hop packets belonging to flow f_2 have prior to n_b and which next-hop is involved after node n_a. The node n_r will code two packets ($P_i^{f_1} \oplus P_j^{f_2}$) to generate coded packets $CP_k^{f_1,f_2}$†. These coded packets can then be transmitted by node n_r to nodes n_a and n_b using a scheduled multicast transmission to the two nodes n_a and n_b respectively.

If one were to use an opportunistic network coding approach akin to Ref. [49] in the above scenario then node n_r would first wait till it received packets (say) $P_1^{f_1}$ and $P_1^{f_2}$ and would then decide that coding is possible as the packets necessary to decode the coded packet ($CP_1^{f_1,f_2} = P_1^{f_1} \oplus P_1^{f_2}$) are available at the next hops n_a and n_b respectively.

As discussed in detail in Sec. 4.1, this would imply that node n_r would have to wait on average for a period T_{mean} as given by Eq. (4.3) before it can transmit the coded packet $CP_1^{f_1,f_2}$. The

* This aspect of local (just prior hop and next-hop being relevant) relevance is very important and will form a cornerstone of our proposed solution SONC.

† We use the notation $CP_k^{f_a,f_b}$ to denote the k^{th} coded packet generated by XORing packets belonging to flow f_a and f_b respectively.

same delay would also be incurred for each and every following coded packet $CP_2^{f_1,f_2}$, $CP_3^{f_1,f_2}$, ..., $CP_n^{f_1,f_2}$. Additionally consider that our intention is to develop solutions for network coding which work in reservation based TDMA WMNs. In such a network — for the required multicast (as well as for the unicast transmissions) — it is only possible to reserve an integral number of minislots for the transmissions in question. This leads to further inefficiency in the above packet-by-packet approach to network coding and transmission of coded packets. To make the above point more clear let us look again at a simplified example. Let us assume that a single minislot is suitable for transmission of B_M bits of data. Let $\mathscr{B}_t(P)$ denote the number of bits required to transmit the packet P including all the necessary header information (e.g. also headers required when P is a coded packet). Now, we have the following three cases (as shown in Eq. (5.2), Eq. (5.2), and Eq. (5.2) respectively) when we want to transmit the coded packet $CP_x^{f_1,f_2}$.

$$\mathscr{B}_t(CP_x^{f_1,f_2}) > i \times B_M \tag{5.2}$$

$$\mathscr{B}_t(CP_x^{f_1,f_2}) < i \times B_M \tag{5.3}$$

$$\mathscr{B}_t(CP_x^{f_1,f_2}) = i \times B_M \tag{5.4}$$

Here, i is an integer representing the number of minislots reserved for the transmission, and $i \times B_M$ represents the number of bits reserved for the transmission in question. For the case in Eq. (5.2), to transmit the coded packet more bits are needed than are reserved, which means that the packet needs to be fragmented, and additional slots for the multicast transmission of the remaining fragment of the coded packet must be reserved, which, in turn further adds to the latency incurred. For the case in Eq. (5.3), we have reserved more slots than actually required for transmitting the data in the coded packet. This means that the remaining bits reserved for transmission are unused, which leads to less efficient usage of bandwidth. The case represented by Eq. (5.4) is the ideal case where we have managed to reserve an exact number of minislots as needed for the multicast transmission of the coded packet, and further the size of the data to be transmitted, $\mathscr{B}_t(CP_x^{f_1,f_2})$, corresponds to bits which can be transmitted in an integral number of minislots. The case of Eq. (5.4) is idealistic and is very unlikely to be the case in real WMNs. If using the above packet-by-packet network coding approach, to avoid additional delay due to fragmentation, the node carrying out network coding (termed as network coding relay for the purpose of our discussion) will try to reserve at least $\lceil \mathscr{B}_t(CP_x^{f_1,f_2})/B_M \rceil$ slots for the multicast transmission of the coded packet $CP_x^{f_1,f_2}$. This, in turn implies that at each such transmission bandwidth equivalent to $((\lceil \mathscr{B}_t(CP_x^{f_1,f_2})/B_M \rceil) \times B_M - \mathscr{B}_t(CP_x^{f_1,f_2}))$ bits will be wasted. Thus, in addition to the overhead of bandwidth reservation and the involved latency, such a contemporary approach to network coding can also be wasteful considering the bandwidth reserved.

Our approach "Stream-Oriented Network Coding" (SONC) breaks away from the contemporary packet-by-packet based approach to network coding and is stream oriented instead. SONC is tailor made to address the issues which arise in deploying network coding in reservation based WMNs and aims to allow efficient network coding deployment in such WMNs without sacrificing the QoS features supported by reservations. We next explain the basic terminology involved and the concept of SONC with reference to the examples in Fig. 5.1. For the sake of the discussion, let us assume that the flow f_1 has an average bandwidth requirement corresponding to 5 minislots per frame, and the flow f_2 has an average bandwidth requirement corresponding to 3 minislots per frame, and that the flow f_3 present in Fig. 5.1(b) has an average bandwidth requirement of 2 minislots.

We have already seen in the above discussion (with reference to Fig. 5.1(a)) that as far as the network coding relay (or just relay) node n_r is considered, it does not matter which path the packets it codes (mixes) together have taken before the immediate previous hops and

their respective next-hops. Thus, as far as coding is considered a network coding relay is only mainly concerned about the immediate previous hops of the packets arriving at the node and the intended next hops. The previous hop of a packet tells the relay node that the packet, in its uncoded form is available at the node for use as an antidote for decoding, the next-hop of a packet is relevant as it enables the node to make an informed choice of the packets it should combine together so that the coded multicast transmission is of value to the intended receivers of the coded packet. A coded packet received at a node is of value if the node is able to decode a packet (or parts thereof[‡]) which was intended for it (either as a final destination, or for relaying further to the next-hop en route to its destination). The condition for a node to be able to decode packet P_i (or parts thereof) from a coded packet have already been explained earlier with the help of Eq. (5.1).

The above insight permits us to design a network coding solution for reservation based WMNs where the network coding decisions are made not on the basis of individual packets arriving at a network coding relay, but are made at the scale of the so called "streams" of packets which are existent at a network coding relay. Furthermore, the reservations for the transmission of coded multicast data is also made to support continuous and stable network coding for the streams chosen to be coded together, and the reservations are not made on a packet by packet basis. The coding itself, is also operating on the order of entire streams of packets and it is not essential to code two (or more) whole packets together to form a coded packet for transmission. We thus see that a fundamental pivot for our solution SONC is the concept of a stream. We will next define what we mean by a stream more precisely.

Definition 5.1.1. ***Stream (local stream)****: We define a local stream or simply stream as the temporal sequence of packets arriving at a relaying node n_r from a given previous hop n_{prev} and intended for forwarding by the relay node n_r to a given next-hop node n_{next}. Thus a stream $\mathscr{S}$ can be uniquely denoted by the tuple $(pred^{n_r}(\mathscr{S}),\ n_r,\ succ^{n_r}(\mathscr{S}),\ \mathbb{P} = \{P_1(\mathscr{S}), P_2(\mathscr{S}), P_3(\mathscr{S}), \ldots\})$, where $\mathbb{P}$ is the ordered set representing the temporal sequence of packets arriving at the relay node n_r from previous hop node given by $pred^{n_r}(\mathscr{S})$ and to be relayed to the next-hop node $succ^{n_r}(\mathscr{S})$. $P_i(\mathscr{S})$ denotes the i^{th} packet belonging to the stream $\mathscr{S}$.*

Note that the Def. (5.1.1) of a stream as used by us always looks at the sequence of packets from the point of view of a given relay node. To make the above notion more clear let us look at the WMN topologies depicted in Fig. 5.1(a), and Fig. 5.1(b). As per our definition for a stream, considering Fig. 5.1(a), there are two streams crossing node n_r. The first stream is composed of the sequence of packets belonging to flow f_1, and the second stream is composed of the sequence of packets belonging to flow f_2. On the other hand, considering Fig. 5.1(b), there are again only two streams crossing node n_r, though we have three different flows in the WMN this time. The first stream is composed of the sequence (temporal) of packets arriving at node n_r belonging to flow f_1. The second stream is composed of the sequence of packets arriving at node n_r which belong to flows f_2 and f_3 respectively.

Thus, considering the example in Fig. 5.1(b) we have the two streams $\mathscr{S}^{f_1}$ and $\mathscr{S}^{f_1,f_2}$. We use $\mathscr{S}^{f_1,\ldots,f_n}_{n_r}$ to denote the stream as in Def. (5.1.1) which is composed of the sequence of packets arriving at the relay node n_r and belonging to flows f_1 to $\ldots$ f_n such that the packets have the same previous hop and the same next hop (as per Def. (5.1.1)). Note, in case probabilistic (e.g. [76, 78, 24, 23]) or multipath routing (e.g. [70, 119]) is being used in the WMN then the next-hops of the packets belonging to the same flow arriving at the relay n_r may be different. In this case packets belonging to the same flow and arriving at a relay node will be associated to different streams as per the definition (5.1.1). Without lack of generality, in this thesis we do not consider multipath routing and for the sake of the remaining discussion we will assume

[‡] This aspect will be explained later when we explain the details of the working of SONC

that only single path routing is used. The designed mechanisms can however be extended in practice to be applicable to probabilistic and multipath routing too.

We can thus see that our definition of a stream as we shall use in this thesis, though similar to the definition of a data stream, audio or video stream as found in Ref. [107], is considerably different in the sense that for packets to belong to a stream only local information (previous hop and next-hop) at a given relay is relevant. Thus, we can identify a temporal sequence of packets as belonging to a stream $\mathscr{S}_{n_r}^{n_{prev},n_{next}}$ when the packets arriving at the relaying node n_r have the same previous hop n_{prev} and will be forwarded to the same next-hop node n_{next}. Returning to our example in Fig. 5.1(b), we have the following streams at node n_r namely:

1. $\mathscr{S}_{n_r}^{n_a,n_b}$: $P_1^{f_1}, P_2^{f_1}, \ldots, P_i^{f_1}, P_{i+1}^{f_1}, \ldots$
2. $\mathscr{S}_{n_r}^{n_b,n_a}$: $P_1^{f_2}, P_2^{f_2}, P_1^{f_3}, P_3^{f_2}, P_2^{f_3}, \ldots, P_i^{f_2}, P_k^{f_3}, P_{k+1}^{f_3}, P_{i+1}^{f_2}, \ldots$

As can be seen above, a stream can consist of an interleaved sequence of packets belonging to different flows. Let $P_i(S)$ denote the i^{th} packet belonging to a stream, where the packets belonging to a stream are assigned increasing indices starting from 1 as per their arrival at the relaying node n_r. Thus, referring to our sample enumeration of packets belonging to stream $\mathscr{S}_{n_r}^{n_b,n_a}$, $P_3(\mathscr{S}_{n_r}^{n_b,n_a}) = P_1^{f_3}$ and $P_4(\mathscr{S}_{n_r}^{n_b,n_a}) = P_3^{f_2}$.

We have now seen in some detail the concept of a stream as defined by us for the purpose of this thesis. As suggested by the name "Stream-Oriented Network Coding" (SONC) the streams are the basic units for network coding. By this we mean that we make informed decisions as to code (mix) packets belonging to certain streams together and then multicast these packets to the respective selected recipients. Additionally, the bandwidth reservations for the multicast coded transmissions are also made for each set of streams we code together (a set of streams coded together at a given relay forms a "network coding constellation"). Thus, reservations are made for a network coding session, and not for transmission of each individual coded transmission. The coding operations are also carried out using the streams as sources for data to code together, and not necessarily on a packet-by-packet basis.

We will now introduce and explain in short the concepts ("network coding constellation", and coding using streams as data sources) which we mentioned above. The details of SONC will be later explained in Sec. 5.2.

Definition 5.1.2. ***Network Coding Constellation****: A network coding constellation at a relay node n_r is uniquely identified by the set of streams $\mathscr{S}_1, \ldots, \mathscr{S}_n$, such that packets (or parts thereof) belonging to these streams can be coded (mixed) together to form a temporal sequence of coded packets for transmission. Thus, the tuple $(n_r,\{\mathscr{S}_1, \ldots, \mathscr{S}_n\})$ uniquely identifies a network coding constellation. Note that the set of streams must be non-empty and have a cardinality of at least two.*

Thus, for each network coding constellation we have a relay node which is central to the constellation, which we term as the network coding relay. In addition we have the end-points of the involved streams. These are nodes which transmit the uncoded (antidote) packets which are basically the previous-hop nodes of the given streams ($pred(\mathscr{S}_i)$), and the nodes which are the next-hop nodes ($succ(\mathscr{S}_i)$) of the given streams $\{\mathscr{S}_i\}$ in the constellation. We use the term "sources" to refer to the set of nodes transmitting the uncoded (antidote) packets for a network coding session corresponding to a given network coding constellation. The term "sinks" is used to refer to the set of nodes which are the recipients of the coded data packets transmitted by the network coding relay.

Note that for a network coding constellation to be useful (i.e. actually result in bandwidth savings via network coding, and allow decoding of new, as yet unreceived packets at the next-hop node) certain criteria need to be met by the set of streams at the given network coding relay. We discuss the details of the criteria later in Sec. 5.3.

Now as per our discussion, SONC will first identify a useful network coding constellation, following which multicast bandwidth reservations will be negotiated with the intended recipients of the coded transmissions for the chosen network coding constellation. Following which the packets belonging to the streams in the network coding constellation will be mixed together giving coded packets which are then transmitted in the slots reserved for these transmissions. We use the term "network coding session" to collectively denote the above phases till the point at which either the network coding constellation ceases to exist or is deemed unsuitable for network coding, or the bandwidth reservations for network coding are cancelled or no longer given. To be more specific we define a network coding session as in Def. (5.1.3).

Definition 5.1.3. ***Network Coding Session****: A network coding session always relates to a given network coding constellation and is a time-limited interchange of information (both control messages and coded and uncoded data messages) between the entities*[§]*. A network coding session is initiated and maintained by the network coding relay which is referred to as the session master.*

A network coding session basically implies a contiguous time span in which the session master initiates actions necessary to deploy SONC for the corresponding network coding constellation (i.e. set up the network coding session). On successful setup of a network coding session, the session master will code packets and transmit the coded packets to the intended recipients till the network coding session is torn down. More details of the lifetime of a network coding session will be discussed later.

We have now introduced the readers to the most important terms which are necessary to understand the concept of stream-oriented network coding proposed by us namely SONC. As discussed earlier, streams form the basic units of operation for SONC, and during the lifetime of a network coding session the network coding relay will mix packets (or parts thereof) belonging to the streams of the corresponding network coding constellation and transmit these to the intended recipients. To explain the above aspect better we will return to our example depicted in Fig. 5.1(b). As shown in the Fig. 5.1(b), we have two streams, $\mathscr{S}_{n_r}^{n_a,n_b}$ and $\mathscr{S}_{n_r}^{n_b,n_a}$, being relayed by the node n_r. Thus, during the lifetime of the network coding session corresponding to the network coding constellation $(n_r,\{\mathscr{S}_{n_r}^{n_a,n_b},\mathscr{S}_{n_r}^{n_b,n_a}\})$ the relay n_r will code and mix consecutive packets (or parts of these) together to generate coded packets which will be transmitted to the intended recipients (in this case n_a and n_b). Thus, for example we will have the node n_r transmitting the sequence of coded packets: …, $P_i(\mathscr{S}_{n_r}^{n_a,n_b}) \oplus P_k(\mathscr{S}_{n_r}^{n_b,n_a})$, $P_{i+1}(\mathscr{S}_{n_r}^{n_a,n_b}) \oplus P_{k+1}(\mathscr{S}_{n_r}^{n_b,n_a})$, $P_{i+2}(\mathscr{S}_{n_r}^{n_a,n_b}) \oplus P_{k+2}(\mathscr{S}_{n_r}^{n_b,n_a})$, …, $P_{i+j}(\mathscr{S}_{n_r}^{n_a,n_b}) \oplus P_{k+t}(\mathscr{S}_{n_r}^{n_b,n_a})$, ….

Note that this sequence is just a simplified example to show how SONC will pick up consecutive packets (or even fragments of packets) from the respective streams to mix together. However, this is not so simple as it seems. In the example in Fig. 5.1(b) we have two streams which can be used for SONC where their individual average data rates sum up to be equal. However, for our example in Fig. 5.1(a) the two streams which can be used for SONC by node n_r have differing average data rates. In such a case it is necessary to carefully choose the number of slots which need to be reserved by SONC for the multicast coded transmissions, also solutions are needed when the streams being combined are asymmetric in terms of data rate, and no data from a given stream is available for coding whereas there is data pending from the other streams. The detailed working of the above operation is explained later.

The main questions which need to be answered when coding using SONC can be summarized as follows:

- Which network coding constellations are suitable for coding, and for which constellations should a network coding session be initiated?

[§] The relay node for the associated network coding constellation, and the set of previous and next-hop nodes for the streams associated with the network coding constellation.

- When do we initiate a network coding session, how to initiate the network coding session, and how to setup the reservations needed for network coding efficiently, and how much bandwidth to reserve for the network coded transmissions?
- How to carry out the actual coding operations, such that the coding efficiently uses the reserved bandwidth, and makes use of pre-existing bandwidth reservations?

In the next section we present the answers to the above vital questions, and thereby also present the details of the working of our solution for network coding in reservation based WMNs: "Stream-Oriented Network Coding".

5.2 Logical Components of the SONC Solution and Architecture

Fig. 5.2 shows the architecture of a node in the WMN which supports SONC as well as CORE[¶]. CORE is our centrally optimized routing extension framework for further enhancing the effectiveness of deploying SONC in reservation based WMNs. SONC is able to operate without CORE too, and does not in any way depend on CORE's mechanisms for its operation. Fig. 5.2 also shows the CORE components for the sake of permitting the readers to understand the interaction of CORE and SONC better when we discuss CORE in Part III of this thesis. We will during the course of the current discussion also highlight how the architecture would change (slightly) in case was CORE was not supported by the nodes in the WMN.

As is seen from Fig. 5.2 logically SONC uses components which are placed at the MAC layer. This choice is logical and ideal as SONC also involves mechanisms to arbitrate medium access control for the multicast transmissions of network coded data. Here, SONC can make use of the data structures and mechanisms provided by the respective MAC layer and extend and adapt these to achieve its goals. The development of SONC here will use the IEEE 802.16 standard as a framework for putting the research in concrete settings and presenting a proof-of-concept for SONC simultaneously. However, the solutions developed for SONC are more generic and allow easy adaptation to other TDMA based WMNs.

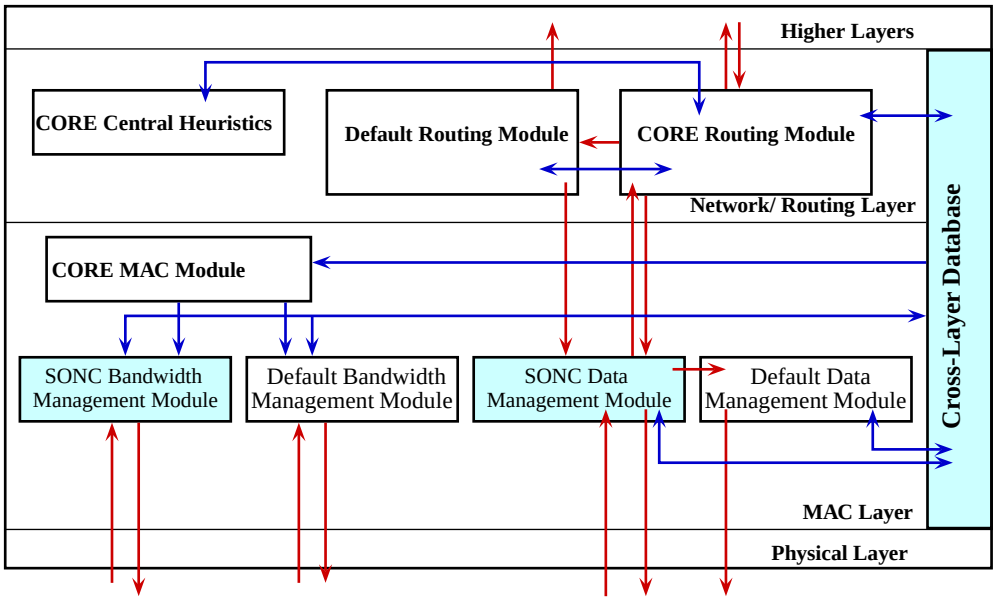
Primary components of SONC are shown shaded in Fig. 5.2. Our deployment of SONC involves the following modules: the **SONC Bandwidth Management Module**, the **SONC Data Management Module**, and the **Cross-Layer Database**. We will next give a short overview of the functionalities provided by each of the above modules for SONC.

The **Cross-Layer Database** permits the convenient exchange of information across different layers in the protocol-stack. However, the same effect can also be reached by other means for transporting information across layers in the protocol stack (see Refs. [105, 106] for a brief overview on the different possibilities for cross-layer information exchange). The major use for the **Cross-Layer Database** for SONC is to identify and associate incoming packets to different streams (as in Def. (5.1.1)). When a node in the WMN receives a packet from a neighbouring node, the node usually knows the previous hop of the packet (as given by the MAC-layer header information usually present in the packets), however, in order for the next-hop of the packet to be known routing information is needed. For packets which are to be relayed by the node, the next-hop is determined usually by the routing algorithm in use. Here, as proposed by us, in Fig. 5.2, the **CORE Routing Module** stores routing information for packets to be handled by the node in the **Cross-Layer Database**, in case the node does not support CORE, then the corresponding information will be stored with the **Cross-Layer Database** by the **Default Routing Module**. In our implementation, SONC (the **SONC Data Management Module**) obtains

[¶] See PART III of the thesis for details about CORE. We show here, for sake of consistency, the architecture of the node assuming that the node in the WMN supports CORE. Only minor changes are needed to allow SONC operate without CORE, we discuss this in depth later.

: Flow of packets (control/data)

: Flow of internal control Information



Protocol Stack of the CORE enabled nodes in the WMN showing Vital Logical SONC Components*

(*Note: vital SONC components are shown shaded)

Figure 5.2: Logical SONC Components and Their Position in the Protocol Stack of a Node Supporting CORE

the next-hop information from the **Cross-Layer Database** by calling a routine provided by the *Cross-Layer Database* and providing the routine with the relevant information available about the packet (i.e. previous hop, source node, destination node). It is also possible to use a solution for SONC which does not rely on interaction with the routing layer for classifying the packets to be relayed to a particular stream. This can be done by hashing the immutable parts of the packet and storing the relevant information about the packet using the hashed value as a key. Once the MAC layer then receives a packet from the upper layer with instructions as to the next-hop as provided by the routing layer, the immutable parts of the packet are hashed again to obtain the key for that packet, and using this key one can then look up the stored information about the packet to get the needed information for classifying the packet as belonging to a particular stream. However, the latter solution would require detailed knowledge of the packet's structure, and hence care would be needed when identifying the immutable parts of the packet. Hence, we use the former solution in our proposal using the **Cross-Layer Database** as an interface between SONC and the corresponding routing modules being used in the WMN. Further, possible applications of the **Cross-Layer Database** are discussed later.

The **SONC Data Management Module** is responsible for handling the packets belonging to the data streams being used for network coding. In addition, it is responsible for the coding of packets belonging to the selected streams, as well as the decoding and defragmentation of received coded packets and packet fragments respectively. The **SONC Data Management Module** is also responsible for maintaining statistical information about the data rates of the individual data streams. Here, it makes use of the routing information provided by the **Cross-Layer Database** as explained earlier. An additional function of the **SONC Data Management Module** is the correct transmission and scheduling of the coded data as well as the uncoded data which will be used by other nodes for decoding the coded data. The **SONC Data Management Module** obtains this information from the **Cross-Layer Database**, where this information is stored by the **SONC Bandwidth Management Module**. Only uncoded and completely defragmented data packets are handed over by the *SONC Data Management Module* to the upper layers for further handling and processing. The statistical data rate information stored by the **SONC Data Management Module** with the **Cross-Layer Database** is used by the **SONC Bandwidth Management Module** to initiate and manage multicast bandwidth reservations for transmission of network coded data, as well as to choose promising network coding constellations for SONC and initiate a network coding session for these streams. The operation of the SONC Data Management Module will be explained in detail when we look in detail at the operational stages of SONC in Sec. 5.3.

The **SONC Bandwidth Management Module** is responsible for maintaining the bandwidth reservations needed for the purposes of deploying SONC at the nodes in the WMN. For this purpose the module supports extended bandwidth reservation handshake protocols which permit the efficient setup of multicast bandwidth reservations for transmission of coded and uncoded packets. It is also responsible for monitoring and observing the streams which are crossing the given node to make informed decisions about the network coding constellations available at the node, and to then choose a selected set of network coding constellations to actively use for network coding (i.e. initiate and maintain a network coding session for the latter). Here, this module makes use of the data provided by the **Cross-Layer Database**, which, in turn obtains the needed data from the **SONC Data Management Module**. The **SONC Bandwidth Management Module** also marks existing unicast bandwidth reservations to denote that these should be used primarily to transmit packets useful for and pertaining to given network coding sessions. Additionally, it also maintains data structures which record the slots reserved for the multicast transmissions corresponding to the individual network coding sessions. The **SONC Bandwidth Management Module** plays a vital role in maintaining the state-machines (see Sec. B.3) corresponding to individual network coding sessions which enables the node to identify the actions it

is to perform at any given point in time. The **SONC Bandwidth Management Module** stores the data structures for bandwidth reservations with the **Cross-Layer Database** in our proposal so that the information can be shared between the different modules at the MAC layer (primarily the **Default Bandwidth Management Module**, the **SONC Bandwidth Management Module**, and the **SONC Data Management Module**) and if needed is also accessible to the modules at the other layers.

To simplify the understanding of SONC, one can identify different operational phases for SONC as listed below:

1. Streams identification and monitoring and identification of promising network coding constellations: In this phase a node which operates as a relay monitors the packets it is relaying to first identify the streams which it is currently handling, and also to then maintain data arrival statistics for each stream such that this data can be meaningfully be used later to identify a subset of the network coding constellations to activate and set up network coding sessions for these.
2. Choosing a (set of) network coding constellation(s) to activate: Based on the data arrival rate statistics for the streams it is handling, the relay then uses the heuristics which we will discuss later to identify a subset of the possible network coding constellations to activate. This choice as to which network coding constellation to activate is made after considering the gain which the relay expects to achieve from activating a particular network coding constellation.
3. Setup of a network coding session: For the network coding constellation which is deemed to be promising enough by the relay a network coding session is activated by the relay. Here the relay uses the advanced and extended handshake (see Sec. 5.3.3) which we have proposed along with the other data structures and mechanisms which allow the relay to efficiently setup the schedule needed for SONC.
4. Operation of SONC for the active network coding sessions: During this phase the relay over time transmits a sequence of coded packets which are formed by mixing uncoded packets received by it belonging to the streams involved in the network coding session. On the other hand the sources and the sink nodes involved in this network coding constellation are in this phase responsible for correctly transmitting the uncoded (antidote) packets and decoding the received coded (poison) packets respectively.
5. Teardown of the established network coding sessions: In this phase the relay which is the session master will prepare to deactivate the network coding session. Hereby the teardown of the session should ensure that the reservations made for network coding are freed, the sources and the sink nodes associated with the network coding session are notified and they decode the remaining packets and then free the resources which had been allocated to the network coding session.

We have now looked at the vital components of SONC in brief and also looked at the location of these in the protocol stack of a node supporting SONC. Furthermore, we presented a short outline of the operational phases of SONC. Details of the individual operational phases of SONC are provided next in Sec. 5.3.

5.3 Phases of SONC Operation

In Sec. 5.2 we looked at the different SONC logical modules involved and their placement in the protocol stack of a node in the WMN. In this section we will look at the individual operational stages of SONC in detail. Note that the operational phases are presented in a logical sequence for a given network coding constellation and its corresponding network coding session. Nodes

in the WMN may have SONC carrying out actions associated with all the different phases of SONC simultaneously for different network coding constellations.

5.3.1 Monitoring of Streams and Detection of Network Coding Constellations

The first step in applying SONC is to identify the streams which are being handled by a node. As discussed in Sec. 5.2 the **SONC Data Management Module** is responsible for handling the data packets when SONC is being deployed. Consider Fig. 5.2, packets arriving at the node from the lower layers are handled by either the bandwidth management modules (**SONC Bandwidth Management Module**, **Default Bandwidth Management Module**) or the **SONC Data Management Module**. The MAC layer hands over all the MAC control messages to one of the two bandwidth management modules based on the packets supported by each module. The data packets as well as control packets from the upper layers are handed over to the data management module (**SONC Data Management Module**)[||]. When the **SONC Data Management Module** receives the packets arriving at the node from the lower layer it queries the **Cross-Layer Database** to find out the next-hop for the packet. The **Cross-Layer Database** uses the information it has from the routing modules at the network layer to answer the query. Now the **SONC Data Management Module** has sufficient information to associate the packet to a unique stream. It will update the statistics maintained by it in the **Cross-Layer Database** to account for the arrival of the number of bits represented by the packet size in the given frame. For each stream the node is handling, the **SONC Data Management Module** maintains a time series representing for each frame the cumulative number of bytes arriving for relaying for the given stream. This data is maintained for a number of previous frames, where the size of this series is given by the parameter *rate-history-size*. This data can be shared across the different bandwidth management modules to estimate bandwidth needed to handle the data arriving at the node. If the stream did not exist previously then a new entry is created for the stream in the **Cross-Layer Database** and statistics associated with the stream are maintained henceforth.

Now, in the context of SONC we have the following two cases:

- Case I: The packet arriving at the node from the lower layers is associated with a stream for which no active network coding session exists.
- Case II: The packet arriving at the node from the lower layers is associated with a stream for which currently an active network coding session exists.

Each of the above cases will entail different handling of the packets arriving at the **SONC Data Management Module**.

In Case I, no active network coding session exists for the packet which has been just received. Hence, there is no need for any additional SONC related operations (decoding) to be carried out. Only the statistics associated with the stream for the packet need to be updated. After updating the statistics, the packet[**] is handed over to the responsible routing module for further processing. In case the packet is to be relayed, the routing module (here the **CORE Routing Module**, in absence of CORE the packets would be directed to the **Default Routing Module**) will choose the appropriate next-hop as per its routing tables and will enqueue the packet with the MAC layer again for relaying to the appropriate next-hop. In Fig. 5.2 we see that the packets from the routing layer are handed over to the **SONC Data Management Module**. In our exam-

[||] In case the node would not support SONC, the MAC control messages would be handed over to the **Default Bandwidth Management Module**, whereas the remaining packets would be handled by the **Default Data Management Module**.

[**] Note, that as is the normal operation, only complete packets are handed over to the upper layer. MAC fragments arriving at the MAC layer are first combined together to a complete packet before handing it over to the upper layer.

ple, in Case I, as no network coding is currently active for the stream this packet is associated to, there are no additional network coding actions (coding) to be carried out before the packet is relayed. Hence, the packet will be handed over by the **SONC Data Management Module** to the **Default Data Management Module** which will then process the scheduling and relaying of the packet as is the case for the standard MAC data packets. In case SONC were not supported by the node then the routing module would have handed over the packet for transmission directly to the **Default Data Management Module**. The **SONC Data Management Module** thus mainly extends the functionality provided by the **Default Data Management Module** to support network coding, and for all other normal data handling which does not involve network coding makes use of the default functions provided by the **Default Data Management Module**.

The Case II listed above is discussed in depth later when we discuss the operation of SONC (see Sec. 5.3.4). Here, we will restrict us to the initial phase of SONC, namely that of identifying streams, associating packets to streams, and identifying network coding constellations. For Case II, the identification of the stream to which a packet belongs to is done using the same procedure as discussed above.

We have now seen in some detail how SONC identifies streams, associates packets with streams, and maintains statistics for individual streams being handled at the node. We have however, not discussed so far how a node should identify network coding constellations given the knowledge that it is currently handling a set of streams $\mathfrak{S}$.

Let us assume that we are considering the relay node n_r. Say we have identified as discussed earlier the set of streams $\mathfrak{S} = \{\mathscr{S}_1, \mathscr{S}_2, \ldots, \mathscr{S}_m\}$ being handled by n_r. As per our definition (5.1.2) of a network coding constellation, we have then theoretically $(2^m - (m+1))$ possible network coding constellations. Assuming that the node n_r has k neighbouring nodes, the theoretical maximum number of network coding constellations which are possible is given by $\left(2^{2\binom{k}{2}} - \left(2\binom{k}{2} + 1\right)\right)$. However, not all of these are meaningful constellations to activate and deploy when using SONC. In this phase of operation SONC filters out the "useless" or infeasible network coding constellations and retains only those which are meaningful.

Let us assume we have a network coding constellation $\mathfrak{N}^{n_r} = (n_r, \{\mathscr{S}_1, \mathscr{S}_2, \ldots, \mathscr{S}_p\})$ at relay n_r. Now as per our definition of the network coding constellation the relay n_r will mix packets[††] from all the streams together to generate coded packets for transmission to the recipients (sinks) for the network coding constellation. Let the coded packet CP_i be given as in Eq. (5.5).

$$CP = P_i(\mathscr{S}_1) \oplus P_j(\mathscr{S}_2) \oplus \cdots \oplus P_r(\mathscr{S}_p) \tag{5.5}$$

Let us define the binary relation $\hat{\in} \subseteq \boldsymbol{P} \times \mathscr{C}$; where $\mathscr{C}$ denotes the set of coded packets, and $\boldsymbol{P}$ denotes the set of uncoded packets. $P_x \hat{\in} C_j$ means that $(P_x, C_j) \in \hat{\in}$. A tuple (P_x, C_j) belongs to $\hat{\in}$ if the uncoded packet P_x is a component of the coded packet C_j, i.e. we have obtained C_j by coding (mixing) packet P_x together with an additional number of packets. Note that it is necessary to have obtained C_j via coding P_x together with a further number of packets. It is not a sufficient condition for (P_x, C_j) to belong to $\hat{\in}$ if C_j can be obtained by coding P_x together with some chosen set of packets. Note that a coded packet is formed by coding together distinct packets. It is not permissible to code a packet with itself (multiple times also) to form a coded packet (see SONC Constraint ***SONC Constraint 3*** below).

Now, for the network coding to be effective the conditions as discussed with reference to Eq. (5.1) should be met. In particular it means that node n_{succ} (a sink node) receiving the coded

[††] For simplifying the discussion assume entire packets are being coded together, in practice parts of a packet can be coded together or packets from certain streams may be skipped altogether if there are no packets from those streams available at the relay for coding. Here, without loss of generality we will assume that a whole packet from each stream in the network coding constellation is used for coding.

packet CP should be able to decode the packet CP such that it is able to obtain in an uncoded form each of the packets in the set $\mathfrak{P}^{n_{succ}}$ given by Eq. (5.6).

$$\mathfrak{P}^{n_{succ}} = \{p : p \hat{\in} CP, \text{ and } p \in S_x, \text{ and } succ^{n_r}(S_x) = n_{succ}, \text{ where } S_x \in \mathfrak{N}^{n_r}\} \tag{5.6}$$

Thus, $\mathfrak{P}^{n_{succ}}$ as represented in Eq. (5.6) basically refers to the set of packets which would be routed normally by the relay n_r to the next-hop node n_{succ}, and which are components of the coded packet CP. S_x is a stream from the constellation $\mathfrak{N}^{n_r}$ for which the relay n_r has generated the coded packet CP. Let $pred^{n_r}(P)$ denote the previous hop taken for packet P prior to its arrival at the node n_r, similarly let $succ^{n_r}(P)$ denote the next-hop for the packet P to which it will be relayed by the node n_r.

We want SONC to work such that not only is it possible for the sink nodes to decode the packets intended for themselves when they receive a coded packet. We additionally want SONC to not introduce any additional latency. This means that in the ideal case all information needed by a node n_{succ} to decode packet CP and obtain the set of packets $\mathfrak{P}^{n_{succ}}$ is present at the node n_{succ} when it receives the coded packet CP.

These constraints help us to identify a set of necessary conditions which must be satisfied by a network coding constellation to be "meaningful"[‡‡]. These conditions are listed next.

Following conditions must be met by the network coding constellation $\mathfrak{N}^{n_r} = (n_r, \{\mathscr{S}_1, \mathscr{S}_2, \ldots, \mathscr{S}_p\})$ for the constellation to be considered a meaningful constellation for SONC.

- ***SONC Constraint 1***: $\forall \mathscr{S}_i, \mathscr{S}_j \in \mathfrak{N}^{n_r}\ (i, j \in [1, p]),$ and $i \neq j$ we have: $succ^{n_r}(\mathscr{S}_i) \neq succ^{n_r}(\mathscr{S}_j)$
- ***SONC Constraint 2***: $\forall n_{succ} \in \{\forall_{p \hat{\in} CP} \cup succ^{n_r}(p)\}, \forall p_x, p_y \in \mathfrak{P}^{n_{succ}}, p_x \neq p_y,$ we have: $S^{n_r}(p_x) \neq S^{n_r}(p_y)$
- ***SONC Constraint 3***: A coded packet CP which is formed by combining the packets as given by the regular expression $CP = (p_i \oplus p_i) + (\oplus p_i) * (\oplus p_k) * (\oplus p_l) * (\oplus p_m) * \cdots (\oplus p_z) *$, $\forall p_i, p_k, p_l, \ldots \in \cup_{\mathscr{S}_t \in \mathfrak{N}^{n_r}} \{\mathscr{S}_t\}$ is not permitted. Where p_i, p_j, p_k, ... are packets.
- ***SONC Constraint 4***: $\forall \mathscr{S}_i, \mathscr{S}_j \in \mathfrak{N}^{n_r},$ and $i \neq j$ we have: $pred^{n_r}(\mathscr{S}_i) \neq pred^{n_r}(\mathscr{S}_j)$.
- ***SONC Constraint 5***: $\forall n_{succ} \in \{\forall_{p \hat{\in} CP} \cup succ^{n_r}(p)\},\ \forall p_x \in \{\{p_k : p_k \hat{\in} CP\} \backslash \mathfrak{P}^{n_{succ}}\}$ we have: either *Case A*: $n_{succ} = pred^{n_r}(p_x)$, or *Case B*: $n_{succ} \in Nbr(pred^{n_r}(p_x))$.
- ***SONC Constraint 6***: The transmission schedule for the packets is such that the following conditions are satisfied:
  1. n_r is able to correctly receive the set of packets $\{p : p \hat{\in} CP\}$, and
  2. each sink node n_{succ} correctly receives the packet CP and the set of packets needed to successfully decode each packet in the set $\mathfrak{P}^{n_{succ}}$.
- ***SONC Constraint 7***: $\forall \mathscr{S}_i \in \mathfrak{N}^{n_r}$ we have: $succ^{n_r}(\mathscr{S}_i) \notin Nbr(pred^{n_r}(\mathscr{S}_i))$

Here, CP denotes the coded packet formed by coding data from the streams belonging to the network coding constellation $\mathfrak{N}^{n_r}$. $\mathfrak{P}^{n_{succ}}$ denotes the set of packets as given by Eq. (5.6). $S^{n_r}(p_x)$ denotes the stream to which packet p_x belongs to when it arrives at the relay node n_r. $Nbr(n)$ denotes the set of neighbours of node n. A formal definition of $Nbr(n)$ is given in Sec. 8.1.1.

SONC Constraint 1 states that for the network coding constellation to be meaningful it is not permissible for two streams in the network coding constellation to have the same node as the next-hop after the relay. ***SONC Constraint 2*** states that at any sink node for the given network coding constellation, the coded packet should not comprise of two packets from the same stream coded with each other and with other packets from other streams. Any given packet is also not

[‡‡] In the sense that the sink node is able to decode correctly the packets intended for itself adding to the information available at the sink node, and without major addition to the latency, when the node receives the coded packet.

to be coded with itself (***SONC Constraint 3***). ***SONC Constraint 4*** states that for a meaningful network coding constellation the packets belonging to any two streams in the constellation should not have the same previous hop node before the relay. ***SONC Constraint 5*** and ***SONC Constraint 6*** basically ensure that at any given sink node the needed antidote packets can be available to allow the sink node to decode the packets meant for itself on receipt of the coded packet. ***SONC Constraint 7*** is not necessary to be satisfied for SONC to be deployed. The aim of adding this constraint is to highlight to network designers the fact that if ***SONC Constraint 7*** is not satisfied then it is better to adapt the routing to route the packets for the respective stream via a single hop from the predecessor node ($pred^{n_r}(\mathscr{S}_i)$) for the stream to the corresponding successor ($succ^{n_r}(\mathscr{S}_i)$) node instead of relaying the packets via the relay node (n_r).

We will now demonstrate the need for each of the above constraints for meaningful application of SONC.

Assume that for a given pair of streams $\mathscr{S}_i$, $\mathscr{S}_j \in \mathfrak{n}^{n_r}$ we have $succ^{n_r}(\mathscr{S}_i) = succ^{n_r}(\mathscr{S}_j)$. Assume that $succ^{n_r}(\mathscr{S}_i) = succ^{n_r}(\mathscr{S}_j) = n_d$ where n_d is a sink node for the network coding constellation. Using our prior notation $\mathfrak{P}^{n_d}$ represents the set of packets which are components of the coded packet CP (coded by the relay node n_r) and which should be successfully decoded at node n_d. Say the set of packets which thus need to be successfully decoded by the node n_d is enumerated as $\mathfrak{P}^{n_d} = \{P_l(\mathscr{S}_i),\ P_m(\mathscr{S}_j), P_t(\mathscr{S}_q), \ldots\}$. We see that the set $\mathfrak{P}^{n_d}$ consists of packet $P_l(\mathscr{S}_i)$ belonging to stream $\mathscr{S}_i$ and packet $P_l(\mathscr{S}_j)$ belonging to stream $\mathscr{S}_j$. Now for being able to correctly decode packet $p_x \hat{\in} CP$, a given node will need the information from each of the packets in the set of packets $\{\{p : p \hat{\in} CP\} \backslash \{p_x\}\}$ (follows directly from the XOR coding used to form CP). Now in our example it means that for n_d to successfully decode $P_l(\mathscr{S}_i)$ it needs to have all the packets in the set $\{p : p \hat{\in} CP\}\{P_l(\mathscr{S}_i)\}$. Which implies that it must have the packet $P_m(\mathscr{S}_j)$. On the other hand in our example the node n_d also needs to decode the packet $P_m(\mathscr{S}_j)$, for which using similar arguments we come to the conclusion that the node n_d must already have the packet $P_l(\mathscr{S}_i)$. But this implies that the node n_d already has the packets $P_l(\mathscr{S}_i)$, and $P_m(\mathscr{S}_j)$ which were to be decoded from the packet CP. Thus it makes no sense to code these packets and transmit them coded together with other packets when the only node supposed to receive the packet (we have assumed that we look into the case of unicast flows only, and thus for each packet there will be just one single next-hop node at the relay n_r) already has these packets. Thus coding of packets from two streams with the same next-hop after the relay node is not gainful. Thus we have justified ***SONC Constraint 1***.

Now let us assume that for a given pair of packets p_x, $p_y \hat{\in} CP$ we have $S^{n_r}(p_x) = S^{n_r}(p_y)$. As both packets ($p_x$, p_y) belong to the same stream both the packets should be correctly decoded by the same sink node, n_i, say. This follows directly from the definition of a stream (Def. 5.1.1). Now we need that the sink node n_i be able to correctly decode the packets p_x and p_y on receiving the coded packet CP. Now using arguments similar to the earlier arguments for justifying ***SONC Constraint 1*** we conclude that to decode p_x correctly the node n_i needs to already have the packet p_y and to be able to correctly decode the packet p_y it needs to already have the packet p_x. This means it already has packets p_x, and p_y and hence, it is not gainful to code these packets together and transmit them to the node n_i which already has these packets. Hence, we see that SONC in this case is not meaningful, therefore for any pair of packets p_x, $p_y \hat{\in} CP$ we need that $S^{n_r}(p_x) \neq S^{n_r}(p_y)$. Hence, we have justified ***SONC Constraint 2***.

Let us assume that we code a given packet p_i together with itself a given number of times and with a set of other packets to get the packet CP. Now as we are assuming the case of unicast flows, and single path routing, the packet p_i should be correctly decoded at the respective sink node, n_s, say. Now as we have coded the packet multiple times (m times say) with itself we assume that we want the node n_s to be able to correctly decode all the $m+1$ instances of the same packet which we have coded together, else we would not code the packet with itself and

just code it with the other distinct packets. This, using arguments similar to that used to justify ***SONC Constraint 1*** and ***SONC Constraint 2*** implies that in order to be able to correctly decode p_i the node n_s should already have the packet p_i. In which case it makes no sense to use the packet p_i as a component of CP and transmit it for decoding at n_s as the node already has packet p_i. In this case SONC will lead to no gain by using p_i to generate CP. Hence, we justify ***SONC Constraint 3***.

From these constraints, ***SONC Constraint 1***, ***SONC Constraint 2***, and ***SONC Constraint 3*** we can derive the condition that for any given sink node n_{succ} Eq. (5.7) must hold.

$$|\mathfrak{P}^{n_{succ}}| \leq 1 \tag{5.7}$$

Assume that for a given pair of streams $\mathscr{S}_i$, $\mathscr{S}_j \in \mathfrak{n}^{n_r}$, and $\mathscr{S}_i \neq \mathscr{S}_j$ we have $pred^{n_r}(\mathscr{S}_i) = pred^{n_r}(\mathscr{S}_j)$. Let $pred^{n_r}(\mathscr{S}_i) = pred^{n_r}(\mathscr{S}_j) = n_s$. Now from our definition of a stream and the given assumptions we have $succ^{n_r}(\mathscr{S}_i) \neq succ^{n_r}(\mathscr{S}_j)$. Let $succ^{n_r}(\mathscr{S}_i) = n_i$ and let $succ^{n_r}(\mathscr{S}_j) = n_j$. Now let $P_l(\mathscr{S}_i)$, and $P_m(\mathscr{S}_j) \hat{\in} CP$. The packet $P_l(\mathscr{S}_i) \in \mathfrak{P}^{n_i}$ and packet $P_m(\mathscr{S}_j) \in \mathfrak{P}^{n_j}$. Using arguments similar to those used for justifying the prior constraints, we see that at node n_i in order to correctly decode packet packet $P_l(\mathscr{S}_i)$ we see that packet $P_m(\mathscr{S}_j)$ is one of the packets which must be already available at node n_i. Similarly to enable node n_j to decode packet $P_m(\mathscr{S}_j)$ it needs to have the packet $P_l(\mathscr{S}_i)$ besides the other packets needed. Let us now look at n_i. It can have the packet $P_m(\mathscr{S}_j)$ either while it is the sender of the packet (let us denote this as case *C1*) or n_i is a neighbour of node n_s and overheard the packet $P_m(\mathscr{S}_j)$ when it was transmitted to the relay n_r (let us denote this as case *C2*). Now case *C1* would imply that $n_i = n_s$ and that a packet is being routed from n_i (n_s) to the relay node n_r and then back to the node n_i (n_s) again, which is absurd and implies that there is a loop in the routing, and that the routing is faulty. So case *C1* cannot be the case assuming that the routing is functional and correct. Now if case *C2* is the state in the network, it implies that $n_i \in Nbr(n_s)$ and so can overhear the packet $P_m(\mathscr{S}_j)$ transmitted by the node n_s and use it (along with the other needed packets) to decode CP to obtain the packet $P_l(\mathscr{S}_i)$ which is intended for itself. But as per our assumption, n_s is also the source of the packet $P_l(\mathscr{S}_i)$, and we have now the fact that $n_i \in Nbr(n_s)$. Here, the packet $P_l(\mathscr{S}_i)$ can be much more efficiently and directly transmitted by node n_s to node n_i and there is no need to relay the packet via node n_r and use the packet for coding. In this case a direct transmission would be much more efficient in terms of number of transmissions (and bandwidth) needed than relaying and network coding. Similar arguments can be presented for the case of node n_j and packet $P_l(\mathscr{S}_j)$. Hence, we see that SONC is not meaningful when we code packets belonging to two streams with the same previous hop node prior to the relay. We have thus justified ***SONC Constraint 4***.

Let n_{succ} be a sink node which needs to correctly decode the packets in the set $\mathfrak{P}^{n_{succ}}$ which are components of the coded packet CP transmitted by the relay node n_r. Now from the coding we are using we know that to enable the sink n_{succ} to decode the packets intended for itself it must have each and every packet in the set $\{\{p_k : p_k \hat{\in} CP\} \backslash \mathfrak{P}^{n_{succ}}\}$. Let us assume that p_x is one such packet from the set $\{\{p_k : p_k \hat{\in} CP\} \backslash \mathfrak{P}^{n_{succ}}\}$. Assume that both ***SONC Constraint 5*** *Case A* and ***SONC Constraint 5*** *Case B* do not hold. Thus we have for packet p_x, $n_{succ} \neq pred^{n_r}(p_x)$ and $n_{succ} \notin Nbr(pred^{n_r}(p_x))$. Now, it is clear from the former discussion that the node n_{succ} does need p_x if it is to be able to successfully decode the packet CP to get the packets intended for itself. However, from our assumptions n_{succ} is neither the previous hop node for packet p_x prior to the relay n_r nor is it a neighbour of the previous hop of the packet p_x prior to the relay. Hence, the only other ways that n_{succ} could have the packet p_x are:

- *Case OnPath*: Node n_{succ} lies on the path taken by the packet p_x to the relay node n_r or,

- *Case NbrPath*: node n_{succ} is a neighbour of a node which lies on the path taken by the packet p_x to the relay node n_r.

We will make this clear with the help of a simple example shown in Fig. 5.3. Here $pred^{n_r}(p_x) = n_p$, and $succ^{n_r}(p_x) = n_s$. As per our definition $p_x \in \mathscr{S}_{n_r}^{n_p,n_s}$.

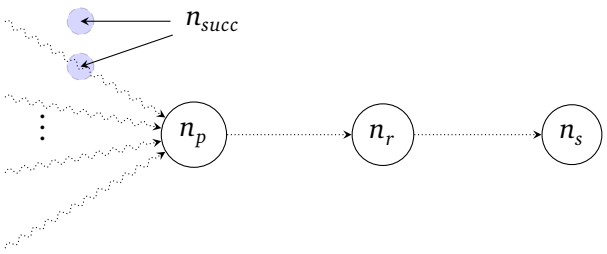


Figure 5.3: Illustrative Example Showing the Relevance of SONC Constraint 5

In Fig. 5.3, the dotted straight arrows show the path taken by the packets belonging to stream $\mathscr{S}_{n_r}^{n_p,n_s}$. The snake-like dotted arrows represent individual paths taken by packets which compose the stream $\mathscr{S}_{n_r}^{n_p,n_s}$ up to the node n_p. n_r is the relay node which should mix packets belonging to the stream $\mathscr{S}_{n_r}^{n_p,n_s}$ with other streams to from the coded packet CP we discussed in our example earlier. n_{succ}, as previously discussed is the node which needs the packet $p_x \in \mathscr{S}_{n_r}^{n_p,n_s}$ in order to be able to decode the packets intended for itself from the packet CP. Now as per our assumptions and the cases which are possible for n_{succ} to be able to receive the packet (as discussed earlier) are shown in Fig. 5.3 where the possible logical positions for n_{succ} are indicated by the two shaded circles. Either n_{succ} lies on the path taken by p_x prior to it's reaching n_p, or n_{succ} is a neighbour of some node which lies on the path taken by the packet p_x. If this is not the case then given our assumptions there is no way that n_{succ} would receive the packet p_x and hence would not be able to decode packet CP successfully to receive packets meant for itself. This would violate our basic requirement for SONC that the sink nodes should be able to correctly decode packets meant for themselves on receipt of the coded packet CP.

Now assuming that we have either *Case OnPath*, or *Case NbrPath* for the given coded packet (CP) in question the node n_{succ} will be able to receive packet p_x which it needs for decoding the packet CP. For simplicity of the discussion let us assume that for the case of the packet CP node n_{succ} has packet p_x and all other packets it needs to be able to successfully decode packets in the set $\mathfrak{P}^{n_{succ}}$. However, for SONC when making the coding decisions, the relay node needs to be sure that the node n_{succ} will indeed be able to receive the packet p_x[§§]. In *Case OnPath* the relay n_r may be able to know that p_x is available at node n_{succ} by overhearing the transmission of the packet p_x by node n_{succ}. However, if this is the case it is in most cases meaningful for the packet to be routed from n_{succ} directly to node n_r as it has arrived at the node after taking a longer path anyway. Such overhearing would not be possible, however, if the data on individual links (multicast, as well as unicast) is encrypted, as is optionally provided by the IEEE 802.16 MeSH mode. The *Case NbrPath* is a bit more complex. Here, node n_r needs to somehow be informed that a packet p_x has been overheard by node n_{succ} although it is on the path taken by the packet p_x. This would imply that additional messages need to be transmitted by the potential sink nodes for a network coding constellation to notify the relay n_r of all the data packets available with themselves. In both cases we see that this means a large overhead for maintaining the correct schedule of transmissions in the WMN (to ensure that the right packets are overheard by the right nodes) and also overhead for the relay when deciding which packets it can code together.

[§§] In addition to the other packets needed, for simplifying the discussion we focus only on packet p_x

Note that in our proposal given a network coding constellation, the relay will code packets picked up from each stream from the network coding constellation to generate the coded packet CP. However, as shown in Fig. 5.3, packets belonging to a given stream may follow a myriad of paths before the previous hop node for the stream. In Fig. 5.3, we see that the packets belonging to the stream $\mathscr{S}_{n_r}^{n_p,n_s}$ take a number of paths before reaching node n_p. As the packets belonging to stream $\mathscr{S}_{n_r}^{n_p,n_s}$ need to be available at node n_{succ} for enabling it to decode the coded packets, either n_{succ} is on the path of the packets prior to n_p or is able to overhear these packets being transmitted along their paths prior to node n_p. This is clearly a very unlikely, and moreover, even if theoretically possible, very difficult to coordinate from the relay node n_r. n_r does not know which paths are taken by a packets belonging to a stream prior to node n_p, it also has no easy means to know and ensure that a packet p_x from the stream $\mathscr{S}_{n_r}^{n_p,n_s}$ is available at node n_{succ}. As to do so would mean that each relay node in the WMN, needs complete knowledge of all the packets available at the sink nodes, and also needs to schedule the transmissions along the paths taken by packets for all the streams belonging to the network coding constellation, if it wants to code packets belonging to the streams in the constellation together. For a WMN of realistic dimensions, it is very difficult to coordinate the required schedules of all the nodes to satisfy the constraints of all different possible relays in the network.

Hence, we propose that if the conditions presented in ***SONC Constraint 5*** (either *Case A*, or *Case B*) are not met then the network coding constellation should be discarded and not used for SONC by the relay node.

SONC Constraint 6 just ensures that given that ***SONC Constraint 5*** is satisfied the relay should verify that the transmission schedule for the nodes involved in the network coding constellation is such that the sink nodes will have the packets needed to ensure correct and delay-free decoding of the packets they are to receive when these nodes receive the coded packet CP.

SONC Constraint 1 – SONC Constraint 5 are fundamental constraints which need to be satisfied by a given network coding constellation in order that the corresponding relay node considers this network coding constellation as a possible constellation for activation. If any of these constraints (1–5) is not satisfied then the network coding constellation will not be considered to be "meaningful" and discarded. The ***SONC Constraint 6*** is used by the relay as a guiding principle during activation of a network coding session for a meaningful network coding constellation.

SONC Constraint 7, as discussed earlier, is not a necessary condition for the deployment of SONC, however we add this constraint to the list of constraints for meaningful SONC constellations to guide network designers planning to use SONC to the fact that if direct transmission is possible between $pred^{n_r}(\mathscr{S}_i)$ and $succ^{n_r}(\mathscr{S}_i)$ for a given stream $\mathscr{S}_i$ in the constellation $\mathfrak{N}$ then this should be preferred and the stream $\mathscr{S}_i$ should be removed from the network coding constellation.

Consider the example shown in Fig. 5.4. Here we have the given network topology as shown in the figure with the routing of packets such that we have the three streams $\mathscr{S}_1$, $\mathscr{S}_2$, and $\mathscr{S}_3$ at the relay node n_r. As can be seen from the figure if the relay picks up a single packet from each stream to code together to form the coded packet CP and if the transmission schedule for the individual packets satisfies ***SONC Constraint 6*** then all the constraints ***SONC Constraint 1 – SONC Constraint 6*** are satisfied and SONC is feasible. For example consider a coded packet $CP = P_i(\mathscr{S}_1) \oplus P_j(\mathscr{S}_2) \oplus P_k(\mathscr{S}_3)$ is transmitted by the relay n_r. Then we see that given that the constraints ***SONC Constraint 1 – SONC Constraint 6*** are satisfied, node n_1 has the packets $P_i(\mathscr{S}_1)$ and $P_j(\mathscr{S}_2)$ it needs to successfully decode CP and retrieve the packet $P_k(\mathscr{S}_3)$ which is intended for node n_1. However, n_1 is a direct neighbour of node n_3 and hence could have been sent the packet directly, or it can possibly also overhear the packet when n_3 transmits it to

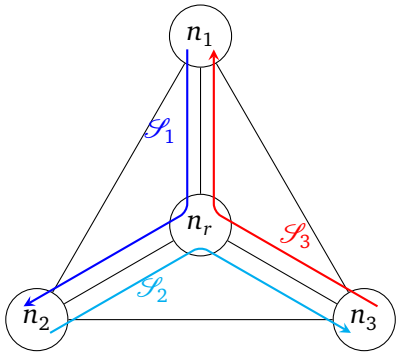


Figure 5.4: Example for a Network Coding Constellation Where SONC is Feasible but not Meaningful

node n_r. Thus, we see that though SONC is feasible it is not necessary, in fact, as seen from this scenario, routing in this case so that SONC is possible is contra productive. Here for the exchange of the three packets which are components of CP we need a total of four transmissions using SONC. The same exchange of information is possible here without using SONC and needs just three transmissions. Thus, we see that although SONC is feasible it is not meaningful if ***SONC Constraint 7*** is not satisfied. Another reason for ***SONC Constraint 7*** is the fact that if we are using SONC then we need to ensure that the transmission schedule is such that all the sink nodes involved in the network coding constellation are able to receive all the packets needed for them to be able to decode the coded packet CP and retrieve the packets intended for the respective nodes.

A network coding constellation $\mathfrak{N}^{n_r}$ is considered "meaningful" if and only if the constraints ***SONC Constraint 1*** – ***SONC Constraint 5*** and ***SONC Constraint 7*** are satisfied for $\mathfrak{N}^{n_r}$. Only such network coding constellations will be considered for activation by SONC where care is taken to ensure that ***SONC Constraint 6*** is also satisfied.

We have now seen in some depth the basic requirements for SONC in general, and discussed how streams are identified at a given relay node. Additionally we have seen the constraints which a network coding constellation needs to satisfy for it to be considered meaningful. For the rest of our discussion, without loss of generality, we will limit ourselves to network coding constellations with just two streams only. We next justify the reasoning for our limitation.

Let $\mathfrak{N}^{n_r} = (n_r, \{\mathscr{S}_1, \mathscr{S}_2, \ldots, \mathscr{S}_m\})$ be a meaningful network coding constellation. Let us define set $Pred = \cup_{i=1\ldots m}\{pred^{n_r}(\mathscr{S}_i)\}$ and the set $Succ = \cup_{i=1\ldots m}\{succ^{n_r}(\mathscr{S}_i)\}$. Now from our SONC constraints we have $|Pred| = m$ and $|Succ| = m$. Note that it is possible that $Pred \cap Succ \neq \emptyset$. For a given node $n_s \in Succ$, let $\mathscr{A}^{\mathfrak{N}^{n_r}}(n_s)$ denote the set $Pred \backslash \{n_s \cup n_{prev}\}$, where $\mathscr{S}_{n_r}^{n_{prev},n_s} \in \mathfrak{N}^{n_r}$. Besides the requirements on the source and destinations for the individual streams in the network coding constellation, the SONC constraints also require that the network topology satisfy certain constraints which stipulate that each sink node in the set $Succ$ be a neighbour of at least $(m-2)$ nodes besides the relay node; i.e. $\forall_{n_s \in Succ}$, we have $:n_s \in \{\cap_{n_p \in \mathscr{A}^{\mathfrak{N}^{n_r}}(n_s)} Nbr(n_p)\}$. This means that for a large m the network topology around the relay will need to be very densely connected (almost fully connected). Additionally for each sink node n_s we need that the sink node be able to correctly receive the uncoded transmissions of packets belonging to the streams in the network coding constellation with source nodes in the set $\mathscr{A}^{\mathfrak{N}^{n_r}(n_s)}$. Looking at the same problem from the perspective of the source nodes for the network coding constellation it means that each source node needs to correctly schedule k-way $(k \geq (m-1))$ multicast transmissions for the uncoded packets belonging to each stream. If $k = 1$ then the multicast reduces to a simple unicast transmission. For a large m the probability of successfully achieving this for any given

source node for the constellation is very low (see discussion in Sec. 4.1.1). Moreover we need to achieve this not just for a single source in the network coding constellation, but for each and every source in the network coding constellation in order to gain from SONC. It is obvious (from the arguments in Sec. 4.1.1) that the probability of successfully achieving this is very small for a large m.

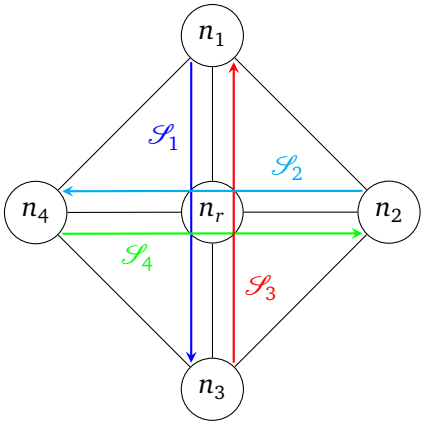


Figure 5.5: Example for Meaningful Network Coding Constellation With Coding Over Four Streams

To make the above discussion more clear let us look at the simple example in Fig. 5.5. Here, at the relay n_r it is possible to deploy SONC for the network coding constellation with the four streams shown in the figure. However, as seen from the figure, it requires the network topology to be very densely connected and also each source to schedule the transmission of the uncoded packets as at least a $(m-1)$-way multicast. In our example $m = 4$, and we can see for example that node n_1 needs to schedule the transmissions of the uncoded packets belonging to stream $\mathscr{S}_1$ as a multicast to be received by nodes n_2, n_4, and node n_r. Similar multicast transmissions need to be successfully scheduled by each of the remaining source nodes for the network coding constellation (i.e. nodes n_2, n_3, and n_4).

Thus, for practical reasons we will limit the remainder of our discussion to SONC for network coding constellations with at most two streams. The presented mechanisms, however, can be easily extended to be applicable to SONC for network coding constellations with more than two streams.

5.3.2 Choosing Network Coding Constellations for Deployment

In the previous section we have seen how streams are identified at the relay node. Further, we have seen how the relay finds out the network coding constellations for which SONC is meaningful. As discussed earlier, without loss of generality, we limit our discussion to the case of SONC for network coding constellations with only two streams. The presented concepts and mechanisms are easily extensible for the SONC with more than two streams.

The aim of this section is to discuss how a relay node n_r decides which network coding constellations it should deploy (activate) and when it should activate these. Here we will also look at the issues which need to be considered when a network coding session is to be established for a given network coding constellation.

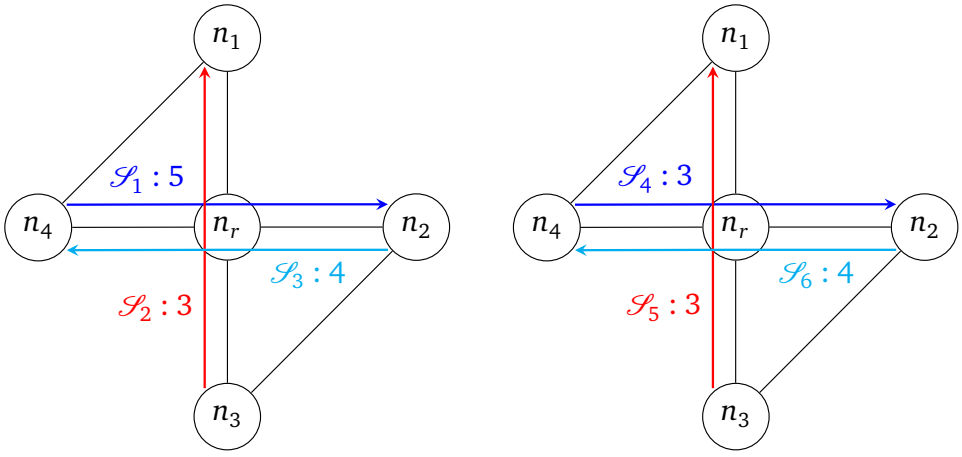
More concretely let us assume that n_r is a relay node which has detected a set of meaningful network coding constellations $\aleph = \{\mathfrak{N}_1^{n_r}, \mathfrak{N}_2^{n_r}, \ldots, \mathfrak{N}_l^{n_r}\}$.

We now need to find means to answer the following questions:

- How do we choose between two network coding constellations for activation, i.e. which constellation to activate earlier when both network coding constellations can be activated, however, when activating one may lead to the other being no longer feasible?
- How do we decide when to activate a network coding constellation $\mathfrak{N}_i^{n_r} \in \aleph$?

Consider the simple network topology shown in the figures Fig. 5.6(a), and Fig. 5.6(b) and the streams $\mathscr{S}_1, \ldots, \mathscr{S}_6$ as shown in the figures. The relay node for all the streams is the node n_r. The numbers following the : in the figures after the stream identifier denote the mean bandwidth demand for individual streams in terms of minislots needed per frame.

Consider the case depicted in Fig. 5.6(a). Here, we have the three streams $\mathscr{S}_1, \mathscr{S}_2$, and $\mathscr{S}_3$. As discussed earlier we will restrict the discussion to network coding constellation with two streams only. Thus at the node n_r we have the set of meaningful network coding constellations $\aleph = \{(n_r, \{\mathscr{S}_1, \mathscr{S}_2\}), (n_r, \{\mathscr{S}_1, \mathscr{S}_3\})\}$. The network coding constellation $(n_r, \{\mathscr{S}_2, \mathscr{S}_3\})$ is not meaningful as per the SONC constraints we discussed earlier.



(a) Case 1 (b) Case 2

Figure 5.6: Sample Topologies with a Number of Streams at Relay n_r

Let $\mathfrak{N}_1^{n_r} = (n_r, \{\mathscr{S}_1, \mathscr{S}_2\})$ and $\mathfrak{N}_2^{n_r} = (n_r, \{\mathscr{S}_1, \mathscr{S}_3\})$. Both these network coding constellations satisfy the constraints ***SONC Constraint 1 – SONC Constraint 5*** and ***SONC Constraint 7*** and thus are meaningful as described earlier and can be considered for activation, i.e. for setting up a network coding session. However, there is a vital difference between these two constellations. The network coding constellation $\mathfrak{N}_1^{n_r}$ requires *overhearing* for the network coding session to be effective when in operation. On the other hand for the network coding constellation $\mathfrak{N}_2^{n_r}$ no overhearing is needed. We use the term overhearing here in this context to highlight the fact that the unicast transmissions of the uncoded data by the source(s) in the network coding constellation to the relay node need also to be overheard (correctly received) by a subset of the sink nodes for the network coding constellation.

For instance, in the case of $\mathfrak{N}_1^{n_r}$, say we have a coded packet $CP = P_i(\mathscr{S}_1) \oplus P_k(\mathscr{S}_2)$ which is generated and transmitted by the relay node when the network coding constellation is active (deployed). To ensure correct decoding of the packets intended for the sink nodes we need that the sink node n_1 be able to overhear the transmission of the uncoded packet $P_i(\mathscr{S}_1)$ from source n_4 to the relay n_r. Similarly, the sink node n_2 needs to overhear the transmission of packet $P_k(\mathscr{S}_2)$ from the source n_3 to the relay node.

In the case of network coding constellation $\mathfrak{N}_2^{n_r}$ no overhearing is needed as the sink nodes are themselves the sources for the packets which are needed at the respective nodes in order for them to be able to decode the packets intended for themselves. For instance, say $CP = P_m(\mathscr{S}_1) \oplus P_n(\mathscr{S}_3)$ is the coded packet which is transmitted by the relay node n_r when the network coding session corresponding to the constellation $\mathfrak{N}_2^{n_r}$ is active. Here, the sink node n_2 should be able to decode and receive the packet $P_m(\mathscr{S}_1)$ on receiving the coded packet CP and the other sink node n_4 should be able to correctly decode the coded packet and obtain the packet $P_n(\mathscr{S}_3)$. Here the node n_2 has itself sent the packet $P_n(\mathscr{S}_3)$ which it needs to be able to decode CP and obtain the packet $P_m(\mathscr{S}_1)$. Similarly, node n_4 is the source for the packet $P_m(\mathscr{S}_1)$ which it needs to decode CP and obtain packet $P_n(\mathscr{S}_3)$.

We can thus see that for network coding constellation where overhearing is needed for the transmissions of one or more source nodes in the constellation the source nodes which need to be overheard need to schedule their unicast transmissions such that overhearing is possible. Effectively we can say that the unicast transmission of the uncoded data packet from the source to the relay node needs to be scheduled as a multicast transmission from the corresponding source node to the relay and the set of nodes which need to overhear the transmission from that source. This is more difficult to schedule than the case where the uncoded transmissions are simple unicast transmissions. Therefore, when choosing one network constellation over the other for deployment priority should be given to the constellations which require no overhearing.

For a given network coding constellation $\mathfrak{N}^{n_r}$, let $overhear(n_i)$ denote the set of nodes (excluding the relay n_r) which need to overhear the unicast transmission of the uncoded packets from the source node n_i to the relay node n_r. The set $overhear(n_i)$ can be computed as given by Eq. (5.8).

$$overhear(n_i) = sinkset(\mathfrak{N}^{n_r}) \backslash \{n_i \cup \{n_k : n_k = succ^{n_r}(\mathscr{S}_t) \text{ and } pred^{n_r}(\mathscr{S}_t) = n_i, \mathscr{S}_t \in \mathfrak{N}^{n_r}\}\} \tag{5.8}$$

$$sinkset(\mathfrak{N}^{n_r}) = \cup_{\mathscr{S}_t \in \mathfrak{N}^{n_r}} succ(\mathscr{S}_t) \tag{5.9}$$

$$sourceset(\mathfrak{N}^{n_r}) = \cup_{\mathscr{S}_t \in \mathfrak{N}^{n_r}} pred(\mathscr{S}_t) \tag{5.10}$$

The set $sinkset(\mathfrak{N}^{n_r})$ represents the set of sink nodes for the network coding constellation $\mathfrak{N}^{n_r}$ at the relay n_r. Similarly, the set $sourceset(\mathfrak{N}^{n_r})$ represents the set of source nodes for the network coding constellation $\mathfrak{N}^{n_r}$ at the relay n_r.

Now given two network coding constellations to choose from a relay node should choose the network coding constellation to activate first where no overhearing is necessary. If both the network coding constellations require overhearing then the network coding constellation for which the number of sources which need to be overheard is lower should be chosen. Considering our example in Fig. 5.6(a) the network coding constellation $\mathfrak{N}_2^{n_r}$ will be preferred to the network coding constellation $\mathfrak{N}_1^{n_r}$.

In general when prioritizing between two network coding constellations for activation, a SONC relay will prefer the constellation for which the value $\sum_{n_i \in sourceset(\mathfrak{N}^{n_r})} |overhear(n_i)|$ *is lower.*

We have now seen that network coding constellations requiring overhearing are more complex to manage and setup (see Sec. 5.3.3 and Sec. 5.3.4 for further details).

However, this is not the only issue which the relay needs to consider when choosing network coding constellations for activation. A relay node n_r is by no means restricted to activating only a single network coding constellation at a time, and in practice can support multiple network

coding sessions simultaneously. Furthermore, a given stream can be a member of multiple network coding constellations. Here, we need to consider some issues which we will discuss next.

An important component of SONC is the reservation of bandwidth for transmission of coded data to a set of sink nodes by the relay node. On the other hand SONC also needs to schedule the uncoded unicast transmissions of each source node involved in the network coding session such that these are correctly received by all the nodes in the *overhear*() set for the source node. This implies that the relay node needs to perform the following actions:

- It needs to reserve a sufficient number of slots for the multicast coded transmissions.
- It needs to ensure that the source nodes are notified when to schedule their uncoded transmissions, and the nodes in the *overhear*() set for the source node know that they should make preparations for being able to correctly overhear the transmissions to the relay node.

The vital issue in both cases is the number of slots which need to be reserved for the multicast transmissions, as well as the number of slots for which the transmissions are scheduled such that overhearing (where needed) of the uncoded data transmissions is possible. The details of the operation will be discussed shortly in Sec. 5.3.3 and Sec. 5.3.4. Here we will just outline the gist of the same process.

SONC aims to amortize the overheads of bandwidth reservation (both the coordination overhead as well as the overhead for correctly maintaining the schedule) via making network coding decisions over entire streams, and reserving bandwidth and scheduling transmissions for chosen network coding constellations. Once this is done packets belonging to the streams in the network coding constellation will be mixed (coded) with each other for the duration of time the session is active. It is in general not feasible to reserve different number of slots for multicast network coded transmissions on a frame-by-frame basis. Similarly, it is very expensive to maintain data structures needed for coordinating the transmissions of uncoded data such that these can be overheard where needed. Hence, SONC aims to reduce this overhead and complexity to a minimum. Thus, when SONC decides to activate a network coding session for a given network coding constellation, it also makes choices about the following:

- It chooses to reserve a fixed number of slots (for a range of frames) for transmissions of the network coded data, and
- the SONC relay coordinates the schedule of the uncoded unicast transmissions such that these are scheduled in a selected range of slots (over a range of frames) where the corresponding nodes in the *overhear*() set are able to overhear these (where needed).

This permits the relay node (which is the session master) to coordinate the reservations and scheduling of the uncoded transmissions at the setup of a network coding session, followed by the coding of the packets (or parts thereof) belonging to the individual streams in the network coding constellation. These reservations need to be updated only when the network coding session is no longer able to make productive use of the reservations. Given this fact SONC needs to determine how many slots should be reserved for the network coding session in question. Here, SONC orients the reservations to the mean data rates of the streams involved. We explain this in brief next with the help of an example.

Consider the sample network coding constellations in Fig. 5.6. Here, we have the network coding constellations $\mathfrak{N}_1^{n_r} = (n_r, \{\mathscr{S}_1, \mathscr{S}_2\})$ and $\mathfrak{N}_2^{n_r} = (n_r, \{\mathscr{S}_1, \mathscr{S}_3\})$ in Fig. 5.6(a), and the network coding constellations $\mathfrak{N}_3^{n_r} = (n_r, \{\mathscr{S}_4, \mathscr{S}_5\})$ and $\mathfrak{N}_4^{n_r} = (n_r, \{\mathscr{S}_4, \mathscr{S}_6\})$ in Fig. 5.6(b). The constellations $\mathfrak{N}_1^{n_r}$ and $\mathfrak{N}_3^{n_r}$ both require overhearing, whereas the network coding constellations $\mathfrak{N}_2^{n_r}$ and $\mathfrak{N}_4^{n_r}$ do not require overhearing. Looking at the two figures Fig. 5.6(a) and Fig. 5.6(b) and the respective streams and the network coding constellations we note that except for the average traffic demands of the involved streams the network coding constellations $\mathfrak{N}_1^{n_r}$ and $\mathfrak{N}_3^{n_r}$ are similar. Likewise the two network coding constellations $\mathfrak{N}_2^{n_r}$ and $\mathfrak{N}_4^{n_r}$ are similar. The

question we now seek to answer is: what implications do the mean data arrival rates of streams belonging to a network coding constellation have for SONC?

The network coding constellations $\mathfrak{N}_1^{n_r}$ and $\mathfrak{N}_2^{n_r}$ share stream $\mathscr{S}_1$. Similarly stream $\mathscr{S}_4$ is common to the network coding constellations $\mathfrak{N}_3^{n_r}$ and $\mathfrak{N}_4^{n_r}$. Consider the case in Fig. 5.6(a). Now if the network coding relay n_r chooses to activate the constellation $\mathfrak{N}_1^{n_r}$ it needs to choose the number of slots it needs to reserve for the multicast coded transmission to the sink nodes (here n_1 and n_2). The mean data arrival rates at the relay n_r for streams $\mathscr{S}_1$ and $\mathscr{S}_2$ are 5 minislots per frame and 3 minislots per frame respectively[¶¶]. For the network coding constellation $\mathfrak{N}_1^{n_r}$ the relay n_r will code data from the streams $\mathscr{S}_1$ and $\mathscr{S}_2$ to form coded packets. Given the mean arrival rates of the two streams the relay can assume that it will have at least data equivalent to 3 minislots from each stream in every frame on average. Now, if the node n_r reserves 3 minislots for the coded multicast transmission in every frame then it can on average fully utilize the reserved bandwidth for transmitting the coded data packets.

Thus, we see that SONC aims to maximize the utilization of the bandwidth reserved for the coded multicast transmissions. Hence, when network coding is activated for a given network coding constellation, the SONC relay reserves the minimum of the mean data rates of all the streams involved for the multicast transmission.

However, in cases where overhearing is needed, SONC additionally needs to ensure that a subset of the packets it will use for generating the coded packet are available at the corresponding sink nodes so that these are able to correctly decode the coded packet and receive the packets meant for themselves. In the above example, assume that the relay node n_r reserves 3 minislots per frame (i.e. minimum of the data rates for the two streams $\mathscr{S}_1$ and $\mathscr{S}_2$) for the multicast coded transmission to the sink nodes n_1 and n_2. Now using SONC when the network coding session for the constellation $\mathfrak{N}_1^{n_r}$ is active the relay node n_r will code data from the two streams $\mathscr{S}_1$ and $\mathscr{S}_2$ to form coded packets which use the space reserved for the multicast transmissions, i.e. three minislots (see Sec. 5.3.4 for more details). In our example for the network coding constellation $\mathfrak{N}_1^{n_r}$ overhearing of the uncoded transmissions from the source nodes is needed. Hence, in order that the relay node be able to code data (equivalent to 3 minislots per frame) from the individual streams together the relay needs to ensure that the source nodes for the individual streams transmit data at at least that data rate and that the data is correctly overheard at the nodes in the respective *overhear()* sets. In our example it means that the node n_4 should transmit data (from stream $\mathscr{S}_1$) in at least 3 minislots per frame to the relay node n_r such that the data is correctly overheard at the node n_1. Similarly the source node n_3 should transmit data (from stream $\mathscr{S}_2$) in at least 3 minislots per frame to node n_r such that these transmissions can be correctly received at node n_2. At the time of setting up the network coding session, the relay node coordinates with all the nodes in the network coding constellation to coordinate slot ranges for the transmission of the unicast transmissions from the sources to itself such that these can be overheard where needed (see Sec. 5.3.3 and Sec. 5.3.4 for details).

Returning to our examples in Fig. 5.6(a) and Fig. 5.6(b) we see that in case of the network coding constellations $\mathfrak{N}_1^{n_r}$ and $\mathfrak{N}_2^{n_r}$ it is possible for the relay node n_r to activate a session for the constellation $\mathfrak{N}_1^{n_r}$ reserving 1 minislot per frame for the multicast and also simultaneously set up a session for the constellation $\mathfrak{N}_2^{n_r}$ reserving 4 minislots per frame for the multicast transmissions. However, for the similar constellation pair $\mathfrak{N}_3^{n_r}$ and $\mathfrak{N}_4^{n_r}$ in Fig. 5.6(b) once the relay n_r has activated a session for the constellation $\mathfrak{N}_4^{n_r}$ using 3 minislots per frame for the multicast coded transmissions, it is no longer possible for it to meaningfully activate a session

[¶¶] To simplify the discussion we present the equivalent mean data rate in minislots per frame, given that *MpF* is the data rate in minislots per frame to get the data rate *BpS* in bits per second we use the following equation: $BpS = (MpF \times B_M)/FDuration$. Where *FDuration* represents the duration of a frame in seconds, and B_M is the number of bits which can be transmitted in a single minislot for the modulation in use.

for the constellation $\mathfrak{N}_3^{n_r}$ as it will no longer have sufficient new uncoded packets belonging to the common stream $\mathscr{S}_4$[***]. Hence, here SONC will activate only one of these two network coding constellations and give preference to the network coding constellation which does not require overhearing.

We have now seen in brief how the session master (relay node) for a network coding constellation chooses the number of slots to reserve for the multicast coded transmissions and also seen that it also needs to coordinate the unicast transmissions from the sources in the network coding constellation in case overhearing of these transmissions is necessary. From this one notices that SONC is especially beneficial when the arrival rates of the individual streams belonging to the network coding constellation are relatively stable and do not vary a lot about the mean data rate. Then, on average the relay will have sufficient uncoded packets from each stream to code together and transmit coded packets in the slots reserved for the purpose. We use some heuristics to classify a stream as either a Constant Bit Rate (CBR) stream or a near CBR stream. Only network coding constellations with at least one of the two streams (as discussed earlier without loss of generality we limit the discussion to network coding constellations with at most two streams) which are either CBR or near CBR will be considered for activation. The process of determining the stability of the data arrival rate for the individual streams is as follows:

1. We first measure the mean data arrival rate for the stream (say stream $\mathscr{S}_i$) in terms of bits (minislots) per frame. To do this we maintain statistics for each stream. This is done by the **SONC Data Management Module** in collaboration with the **Cross-Layer Database** (see Sec. 5.2 and Sec. 5.3.4). Let $\mu_{\mathscr{S}_i}$ denote the mean data arrival rate for a stream measured over a window of the past few frames (for our evaluations we used a window size of 128 frames, i.e. 128 past samples (total bits arriving per frame) were used to compute the mean $\mu_{\mathscr{S}_i}$). Let $B_{n_r}^{\mathscr{S}_i}(t)$ denote the total amount of bits arriving at relay node n_r for stream $\mathscr{S}_i$ in frame t then as discussed we get the following equation for $\mu_{\mathscr{S}_i}$.

$$\mu_{\mathscr{S}_i} = \frac{\sum_{k=0}^{M-1} B_{n_r}^{\mathscr{S}_i}(t-k)}{M} \tag{5.11}$$

   Where we set $M = 128$ for our deployment scenario. If the value $\mu_{\mathscr{S}_i}$ is very small (close to 0) then we consider the stream to be non-stable and non-interesting for network coding, as the data rate is not sufficient or we do not have sufficient samples for the stream to fill the window leading to a very low average data rate.
2. The relay also measures the sample variance (see Ref. [43]) for the data arrival rate for each stream. Let $s_{\mathscr{S}_i}$ denote the sample standard deviation (see Ref. [43]) in the arrival rate for the stream $\mathscr{S}_i$. We can then compute for our scenario the sample standard deviation as follows:

$$s_{\mathscr{S}_i} = \sqrt{\frac{1}{M-1}\sum_{k=0}^{M-1}(B_{n_r}^{\mathscr{S}_i}(t-k) - \mu_{\mathscr{S}_i})^2} \tag{5.12}$$

3. For each stream the relay will also compute the minimum value for the variable $d_{\mathscr{S}_i}$ (≥ 0) such that for the window being considered at time frame t, for at least the selected percentage (given by variable p_d) of samples ($B_{n_r}^{\mathscr{S}_i}(t)$) the condition $|B_{n_r}^{\mathscr{S}_i}(t-k) - \mu_{\mathscr{S}_i}| \leq d_{\mathscr{S}_i}$ holds. Algo. (5) in Appendix B gives the pseudocode for the computation of $d_{\mathscr{S}_i}$ using the procedure $algMinD(B_{n_r}^{\mathscr{S}_i}, \mu_{\mathscr{S}_i}, p_d)$.

[***] Conceptually it may be possible for n_r to use the packets from the stream $\mathscr{S}_4$ twice, i.e. for coding for constellation $\mathfrak{N}_3^{n_r}$ and $\mathfrak{N}_4^{n_r}$. However, it brings no further benefit from network coding as the node n_2 will now receive packets for stream $\mathscr{S}_4$ twice (provided it can correctly decode the coded packets) and only increases the overhead for maintenance of the network coding sessions at n_r

4. Having computed the above values we assume the data arrival rates for the streams belonging to a network coding $\mathfrak{N}_j^{n_r}$ constellation to be stable and suitable for SONC if the following statements hold.

$$\forall \mathscr{S}_i \in \mathfrak{N}_j^{n_r} : \mu_{\mathscr{S}_i} \geq thresholdrate \tag{5.13}$$

$$\exists \mathscr{S}_i \in \mathfrak{N}_j^{n_r} \; s.t. \; q_{cv} \leq t_{cv} \tag{5.14}$$

$$\exists \mathscr{S}_i \in \mathfrak{N}_j^{n_r} \; s.t. \; q_{dstr} \leq t_{dstr} \tag{5.15}$$

Here, the term $q_{cv} = s_{\mathscr{S}_i}/\mu_{\mathscr{S}_i}$ is the coefficient of variation (see Ref. [43]) and gives a measure of the variation in the measured quantity. The term $q_{dstr} = d_{\mathscr{S}_i}/\mu_{\mathscr{S}_i}$ gives a comparison of the magnitude of $d_{\mathscr{S}_i}$ to the magnitude of $\mu_{\mathscr{S}_i}$. If q_{dstr} is close to zero then it means that we have p_d percent of the samples in the current measurement window for the data arrival rate lying very close to the mean data arrival rate and hence the stream can be considered to be near CBR. The thresholds t_{cv} and t_{dstr} can be chosen by the system designer based on the degree to which CBR flows only should be considered for network coding sessions. For our experiments we used the values $t_{cv} = 0.5$, $t_{dstr} = 0.2$, and $p_d = 85$. The variable *thresholdrate* refers to the minimum mean data arrival rate for a stream to be considered for network coding. This parameter can also be chosen suitably by the system designer. For our experiments the value of *thresholdrate* was chosen to be equivalent to one minislot per frame. SONC will only choose network coding constellations for activation for which all the above discussed points are valid and a stream in the network coding constellation can be classified as having a stable data arrival rate.

We have so far seen that SONC will use the mean data rate of the streams in a network coding session to decide how many slots to reserve for the coded multicast transmissions to the sink nodes in the constellation. Additionally, we note that SONC will prioritize for activation network coding constellations with a lower value for $\sum_{n_i \in sourceset(\mathfrak{N}^{n_r})} |overhear(n_i)|$. *We have also seen that SONC will consider only such network coding constellations for activation for which the streams can be deemed to have stable data arrival rates at the relay (see Eq. (5.13), Eq. (5.14) and Eq. (5.15))*.

Another important factor which the relay needs to consider when activating a network coding constellation is the expected gain from activating the network coding session. Here, by gain we mean some benefit to the network WMN in terms of bandwidth savings. One way to measure the benefits from the deployment of network coding is to compare the number of transmissions needed in the network with and without network coding and so measure the network coding gain. This, for example is done by the authors in Ref. [49]. They define the network coding gain as the ratio of the number of transmissions required by a non-coding approach to the number of transmissions using COPE in order to deliver the same number of packets. This is termed by the authors as Coding Gain, there they also introduce a so called Coding+MAC gain. Here, additionally MAC layer issues typical to the IEEE 802.11 standard are considered whereby the nodes are able to better utilize their fair share of the bandwidth using network coding.

Such approaches are not suitable for measuring the gain due to SONC. Here, we use reservations to explicitly reserve bandwidth (slots) for the transmissions (both coded as well as uncoded transmissions). Every reservation leads to some nodes in the WMN being blocked from either transmitting data, or receiving data or both in order to ensure the conflict freedom of the schedule (see Sec. 3.3 for a detailed discussion of scheduling in the IEEE 802.16 Mesh Mode). Even if data packets are not transmitted in the reserved slots this leads to a loss of the capability of nodes in the WMN to communicate with each other. Further, not all transmissions can be weighted equally, it is obvious that transmissions scheduled in densely connected areas of the

WMN will cause a higher number of nodes to be blocked via the reservations for the transmissions as for the case where the same number of transmissions are carried out at a node on the edge of the WMN. Hence, we need to develop a means to measure the impact of the schedule (with and without network coding) on the ability of the WMN to accommodate and schedule additional reservations in the network.

To meet this goal we have proposed in Refs. [80, 82] a metric which we term as the *scheduling degree of freedom*, or simply *degree of freedom* which is available to the WMN in each slot in the system. To avoid breaking the flow of discussion here we present the details of the degree of freedom metric we propose in Appendix A. Readers who are not familiar with our work will find sufficient details in Appendix A to allow them to follow the rest of the discussion.

A node can use the degree of freedom metric we propose in Appendix A to compute in advance if deploying network coding constellation is gainful or not. Here, it can measure the costs of the involved transmissions when using network coding and when not using network coding and compare these two. As discussed in the Appendix A this is especially useful when deciding whether to activate a network coding session requiring overhearing as for such network coding constellations it is not always beneficial to activate coding sessions.

Thus, to summarize in brief a relay node will take note of the following issues before activating a network coding session for a meaningful constellation[†††]:

- SONC will aim to reserve bandwidth corresponding to the minimum mean arrival rate of the streams involved in a network coding constellation for network coding multicast. If the mean arrival rates for the streams in a network coding constellation are below a selected threshold the constellation will not be activated.
- SONC will give priority to constellations with a lower value of $\sum_{n_i \in sourceset(\mathfrak{N}^{n_r})} |overhear(n_i)|$ for activation.
- Only network coding constellations with steady data arrival rates as specified by equations Eq. (5.13), Eq. (5.14) and Eq. (5.15) will be chosen for activation.
- Before activating a network coding constellation with computed values for the number of slots to be reserved for the multicast transmission the relay node will compute the gain expected by using network coding using the degree of freedom metric defined by us in Appendix A. Only if positive gains can be obtained will a network coding session activated.

Having discussed how to choose which network coding constellations should be chosen for activation, we next present the mechanisms designed by us for efficiently activating a network coding session for a given network coding constellation.

5.3.3 Network Coding Session Setup

We restrict the discussion of a network coding session setup to the case of network coding constellations with a maximum of two streams. However, as discussed earlier the presented solutions do not lose their generality and can also be easily extended to the case of network coding constellations with more than two streams. Only such network coding constellations will be considered for activating a network coding session which are meaningful (see Sec. 5.3.1) and which meet all the criteria for activation as discussed in Sec. 5.3.2.

Consider a network coding constellation $\mathfrak{N}^{n_r} = (n_r, \{\mathscr{S}_1, \mathscr{S}_2\})$. Here, n_r is the relay node which wants to activate a network coding session for the network coding constellation $\mathfrak{N}^{n_r}$. $\mathscr{S}_1$ and $\mathscr{S}_2$

[†††] We only need to select from meaningful network coding constellations as discussed in Sec. 5.3.1

are the two streams as seen at n_r which comprise this network coding constellation. Let n_{s_1} and n_{s_2} be the two sink nodes for the streams $\mathscr{S}_1$ and $\mathscr{S}_2$ respectively.

When activating a network coding session for $\mathfrak{N}^{n_r}$ the node n_r needs to perform the following tasks:

- Setup bandwidth reservations for the network coded multicast transmission from n_r to the nodes in the set $sinkset(\mathfrak{N}^{n_r})$.
- Setup new bandwidth reservation or initiate the management of the existing bandwidth reservations for the unicast uncoded transmissions relevant to the network coding constellation $\mathfrak{N}^{n_r}$ from the nodes in the set $sourceset(\mathfrak{N}^{n_r})$ to n_r.
- Initiate handshake for the bandwidth reservations above.
- Notify the nodes involved in the network coding constellation about the activation of the network coding session for the network coding constellation.

The node n_r aims to achieve the above goals considering the following criteria:

- The time duration needed to activate a network coding session should be kept to a minimum. Here, means are needed to get the involved nodes to agree to the bandwidth reservations soon, without long delays due to the handshake.
- The bandwidth reservations should ensure collision free schedules for the packets such that the data needed for network coding is received correctly by the nodes involved in the network coding session.

In order to achieve the above goals we design the handshake process for activation of a network coding session to be a two-phase process. The phases of the network coding session activation handshake are:

1. Phase I: In this phase the relay node n_r will carry out a handshake process in order to negotiate the slots to be used for the network coding session (both for the coded transmissions as well as for the corresponding uncoded transmissions). No bandwidth reservations are actually notified to the nodes in the WMN in this phase. The messages for this part of the handshake are exchanged by the nodes involved using slots reserved in the data subframe.
2. Phase II: In this phase the bandwidth reservations agreed upon in Phase I are initiated by an extended handshake (similar to the three-way handshake for distributed scheduling as discussed in Sec. 3.3). The message exchange in this phase takes place in the control subframe to ensure that all the neighbouring nodes are notified of the schedule agreed upon by the relay and the source and sink nodes for the network coding session.

Fig. 5.7 shows the sequence of message exchange for a successful handshake for setting up a network coding session. We next discuss the individual steps in this process in detail.

The event NC_Opp denotes the time when the relay n_r detects that the network coding constellation satisfies the necessary conditions for activation. It then initiates the network coding session initialization handshake shown in Fig. 5.7. The handshake takes place between the relay n_r and the two sink nodes n_{s_1} and n_{s_2}. However, the activate messages are also sent to the source nodes for the network coding constellation (this is not shown in Fig. 5.7 to retain clarity of the presentation).

As discussed earlier the handshake consists of two phases with the first phase involving an exchange of messages in the data subframe and the second phase involving an exchange of messages in the control subframe. In Fig. 5.7 messages sent in the data subframe are shown using solid arrows. Messages exchanges in the control subframe (second phase of the handshake) are shown using dashed arrows. Explain first the aims, to reduce the control latency, explain also how the handshake is done, how to finish the handshake fast and get to an agreement on the

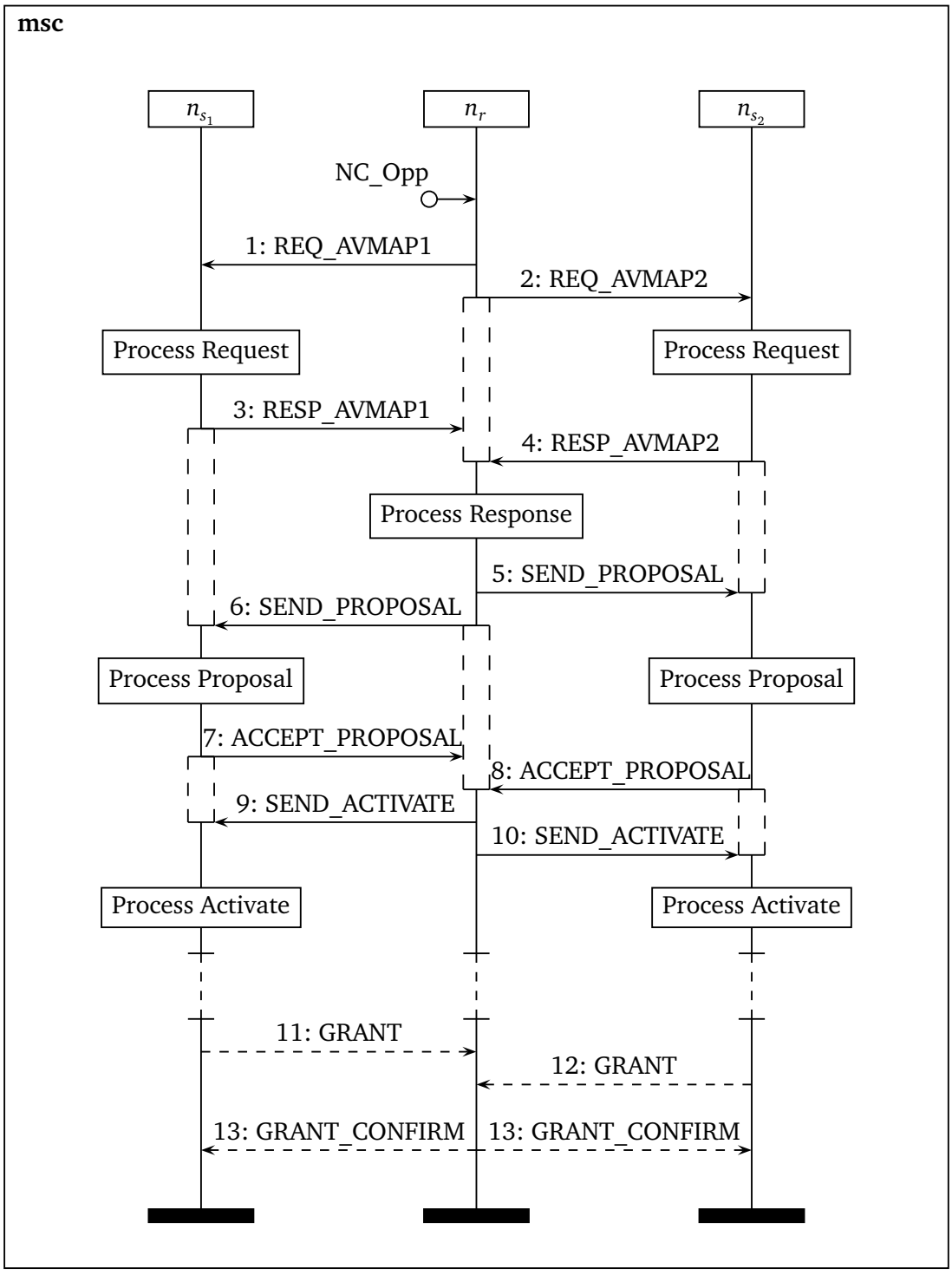


Figure 5.7: Handshake for Initializing a Network Coding Session for $\mathfrak{N}^{n_r}$

sink nodes for the network coding constellation, also how to coordinate in case of multicast is needed for the unicast.

Looking at Fig. 5.7 we see that the first step taken by the node n_r is to send request messages (REQ_AVMAP1, REQ_AVMAP2) to the two sink nodes. The intention of these messages is to obtain from the sink nodes slot ranges (availabilities, see discussion in Chap. 3) in which these nodes can successfully schedule reception of data transmitted by the node n_r, i.e. those slots

with status *rav* or *av*. The sink nodes receiving the request will then process it and generate a response message indicating these suitable slot ranges.

Additionally, for the case of network coding constellations requiring overhearing the request message also carries information about slot ranges where the individual sink nodes should verify if they can overhear messages from the respective source nodes which need to be overheard. Let us consider the simple example in Fig. 5.8 to understand this aspect better.

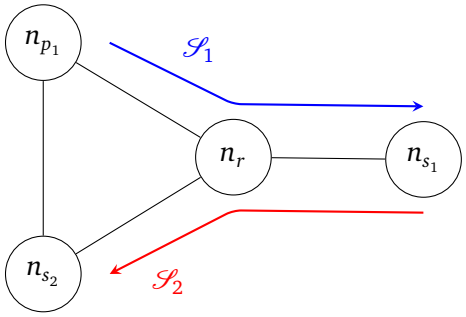


Figure 5.8: Sample Topology for $\mathfrak{N}^{n_r}$

Assume that for the network coding constellation in our example we have the topology and the streams as shown in Fig. 5.8. Now, here the unicast transmissions from n_{p_1} to the relay node n_r for stream $\mathfrak{N}^{n_r}$ need to be overheard at the sink node n_{s_2} in order to permit this sink to to correctly decode the packets intended for itself on receiving a coded packet from the relay. Thus, in the request message in the handshake, the relay n_r in addition to requesting n_{s_2} to identify the slots suitable for reception also notifies it of the ranges of slots which n_{p_1} has reserved for the unicast transmission to n_r with large persistence. By large persistence we restrict ourselves to slots which are reserved with persistence Per_∞ or at least Per_{128}. The reason for this restriction is that once we negotiate ranges where the sink node takes measures to overhear the transmissions from node n_{p_1} we should be able to profit from it for a large number of frames. Where possible and sufficient reservations with Per_∞ are available for the transmissions from the node n_{p_1} to n_r these are to be preferred over reservations with lower persistence. Let *UCST* denote such a range of slots. Then with the request message the relay n_r will also send the range *UCST* to the sink node n_{s_2}. The sink node in this case replies to the request message by specifying in its response a range of slots where it can schedule reception of data transmissions (similar to the case without overhearing). In addition it also checks the current reservations and the status of the slots in the range *UCST* to see if these would be suitable for reception of data if there were no transmission from node n_{p_1} already scheduled (the transmission between n_{p_1} and n_r). If reception is thereby permissible as per the slot status in the range *UCST* or in a subset of the range then this range of slots is also identified and notified to the node n_r along with the response message. The relay node then knows which slot ranges from those reserved by n_{p_1} for transmission to itself are also suitable for overhearing these transmissions at n_{s_2}. The node n_r also knows which slots are suitable at n_{s_2} for overhearing.

We have thus seen the contents of the request messages as well as the main processing in the Process Request action block at the sink nodes. Once the response to the request has been received from both the sink nodes at the relay node it processes the response to the request. The relay looks at the response to its request from the individual sink nodes to see if they have a common range of slots available for reception of data. If this is the case then the relay checks its own data structures to see if these slots are available for transmission at the relay. If this is the case then the relay has successfully found a range of slots which are suitable for scheduling the multicast coded transmission. If overhearing is needed for activation of the network coding

constellation then the relay will additionally check to see if sufficient slots could be found which are also suitable for overhearing at the corresponding sink nodes. Let us assume that this is the case. The relay has then successfully found ranges of slots for its network coded transmission to the sink nodes as well as for overhearing where needed. Hence, the relay now needs to initiate the actual reservation of the slots. So far no concrete reservation of slots is undertaken by any of the nodes involved. Neither are the slot ranges locally temporarily blocked at the sink nodes or at the relay node. This is because so far the sink nodes are not sure which range of slots will be finally chosen by the relay (which is the session master) for the network coded multicast transmission as well as which range of slots will be identified for overhearing.

This is done next by the relay node by sending the proposal message (SEND_PROPOSAL) shown in Fig. 5.7. The proposal message tells the sink nodes i) the range of slots which has been selected for the network coding transmission, and ii) the range of slots where the source nodes which are to be overheard will be transmitting packets from the stream belonging to the constellation, and which should be overheard by the corresponding sink node and stored locally. The sink nodes on receiving the proposal will once again check to see if the slots indicated by the proposal from the relay are still suitable for scheduling the desired actions (i.e. overhearing, as well as the multicast coded reception). If this is the case then, as shown in Fig. 5.7 each sink will respond to the proposal by sending an accept (ACCEPT_PROPOSAL) of the proposal to the relay nodes. Starting at this time the indicated slots will be locally blocked by the involved sink nodes and will not be granted for other parallel reservation handshakes. Once, the relay has collected the accept message (ACCEPT_PROPOSAL) from both the sink nodes it will send activate messages (SEND_ACTIVATE) to the sink nodes indicating successful completion of the first phase of the SONC session initialization handshake. The activate message will be sent to the source nodes for the constellation too, informing these of the slot ranges chosen for overhearing and the stream for which these ranges of slots should be used (for the unicast uncoded transmissions to the relay). Further discussion of the latter aspect is given in Sec. 5.3.4.

The activate messages on reception will be processed by the individual sink nodes to generate grants for the slots to be reserved for the multicast transmission from the relay. Further, if ranges of slots are to be overheard then the sink node which needs to overhear transmissions will generate what we call *pseudogrants* for these slots. Both the grant messages as well as the pseudogrant messages will be transmitted by the sink nodes in the control subframe at the earliest opportunity that these nodes have. As we recollect from Chap. 3 the aim of the grant messages is to indicate to the neighbours of the node emitting these that it wishes to receive data in these slots and hence they may not schedule transmission of data in those slots. The aim of the pseudogrants (for the range of slots to be overheard) is the same. Here, the only difference being that the neighbours of the node issuing a pseudogrant will not block the slots for transmission as was the case for a grant, but mark these internally as not to be used for scheduling transmissions. However, if no other slot ranges are available then as a last resort such slots may be used for transmissions. We use reserved bits in the MSH-DSCH message in the standard to distinguish between pseudogrants and grants, and will not discuss this here in detail. Also the MeSH mode uses reservations per link, we use a link id for the multicast transmission which is also notified to the nodes involved (i.e. the sink nodes) in the activate message. Thus, instead of a unicast link id we have a multicast link id in the MSH-DSCH messages and can thereby use the standard's messages for realizing our solution without requiring any additional modifications to this message. This, is however, not of primary interest here, and other realizations of the same are possible.

The transmission of the grant messages is shown in Fig. 5.7. Both the grants will be for the same multicast link and for exactly the same range of slots as indicated by the proposal

received from the relay. Pseudogrants will be transmitted by the sink which needs to overhear transmissions along with the grant messages. These are not shown explicitly in Fig. 5.7 to avoid confusion. The relay will wait till it receives grants from both the sink nodes before issuing a grant confirmation corresponding to the grant. This corresponds to the last two messages exchanged in the normal three-way handshake for distributed scheduling in the IEEE MeSH mode (i.e. grant followed by a grant confirmation). The grant confirmation indicates to the neighbours of the relay that it has scheduled a (multicast) transmission in the slots indicated by the grant confirm. The grants and pseudogrants indicate scheduling of reception of data as discussed earlier. Thus, the second phase of the SONC session initialization handshake uses messages transmitted in the control subframe with the aim of scheduling the transmissions planned in the first phase of the handshake and reserving slots for these transmissions.

Here, we discussed with the help of Fig. 5.7 a successful SONC session initialization handshake. Such a handshake may however also fail. We discuss representative failures of the handshake in Appendix B.2. To enable the relay and the sink nodes to react appropriately to the messages received so far in the handshake these maintain state-machines for each session being activated. A discussion of these state-machines is given in the Appendix B.3.

5.3.4 SONC in Operation: Coding Over Streams

We have seen now that the session master (relay node) will use the SONC session initialization handshake described in Sec. 5.3.3 to activate a network coding session. Using the handshake the relay has set up the reservations needed for the multicast network coded transmissions to the sink node. The sink nodes on the other hand have registered which ranges of slots are to be used for overhearing (if needed) and have notified their neighbourhood about this using the pseudogrants. The source nodes have also been notified about the ranges of slots in which they should transmit uncoded data belonging to the streams for the active network coding constellation to the relay such that the nodes needed to overhear these transmissions can do so. In this section we will look at the operations performed by the individual participants when SONC is active (i.e. a network coding session has been set up and is active). The nodes participating in a network coding session are the relay node, the sink nodes and the source nodes.

We will next look at the operations performed by each of these participants. To simplify the explanation and improve clarity of the explanation we will explain the operations of the individual participants in the network coding session with the help of the example in Fig. 5.9.

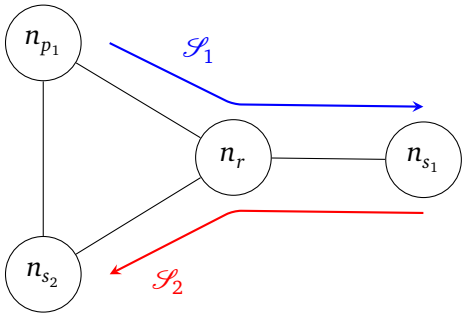


Figure 5.9: Sample Topology for Explanation of Operation of SONC

Let $\mathfrak{N}^{n_r} = (n_r, \{\mathscr{S}_1, \mathscr{S}_2\})$ be a network coding constellation with the streams and the relay node as shown in Fig. 5.9. This is a network coding constellation needing overhearing. n_r is the relay for the network coding session, the set $\{n_{p_1}, n_{s_1}\}$ is the set of source nodes and the set $\{n_{s_1}, n_{s_2}\}$

is the set of sink nodes for the network coding session. The uncoded data belonging to stream $\mathscr{S}_1$ transmitted by node n_{p_1} to node n_r needs to be correctly overheard at node n_{s_2}. We assume that *resv* bits (slots corresponding to *resv*) have been reserved for the coded transmission by node n_r to the two sink nodes n_{s_1} and n_{s_2} in each frame.

Operation at Relay

The relay node needs to code the uncoded packets belonging to the streams from the network coding session and transmit these to the sink nodes. To do this it must keep track of the uncoded packets arriving at the relay from the source nodes of a constellation to find out which packets may be used for coding. Once the relay has identified packets which it may code together it must then efficiently generate coded data such that the reserved multicast bandwidth is efficiently utilized.

Consider the example in Fig. 5.9 with the network coding constellation $\mathfrak{N}^{n_r}$ as discussed above. When the network coding session for this constellation is active the relay will mix information (packets, or parts thereof) from stream $\mathscr{S}_1$ with information from stream $\mathscr{S}_2$ to generate coded packets. Fig. 5.10 shows how the relay will do this.

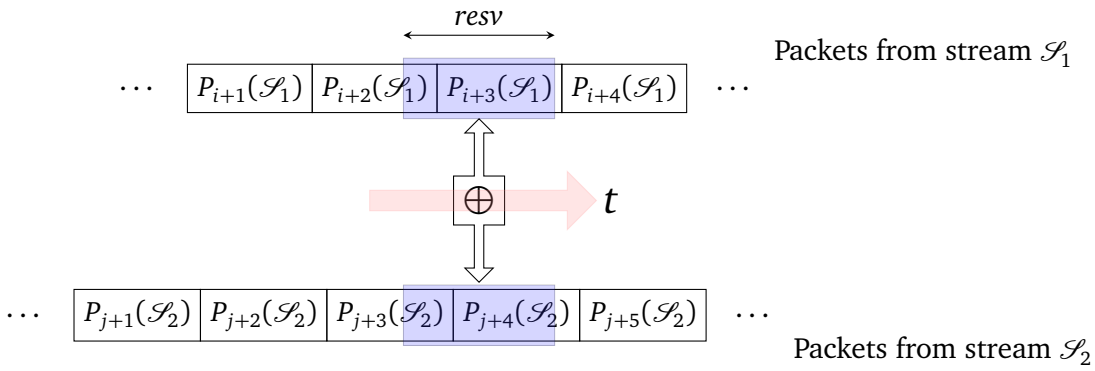


Figure 5.10: Coding Over Streams

Logically the relay will consider the sequence of uncoded packets arriving at the relay for each individual stream as a continuous "tape" of data. Thus, as shown in Fig. 5.10 we will have one such logical tape of data for stream $\mathscr{S}_1$ and one tape of data for stream $\mathscr{S}_2$. On each tape the relay will maintain a current window of size *resv* bits of information. When the relay needs to schedule transmissions for the multicast transmission, it will pick up *resv* bits from each tape of data and mix this together to create the coded data packet which is transmitted in the slots reserved for the multicast transmission. The current window will then move forward over the individual tapes to cover the next *resv* bits of data which have not yet been mixed.

To explain the basic concept above we made a number of simplifications in the discussion. In practice the size of data picked up from each logical tape (corresponding to each stream) will be slightly less than the *resv* bits which have been reserved for the network coding transmission to account for the overhead of headers (for the MAC layer as well as for network coding). To enable the nodes receiving the coded data to identify which data has been coded together we put a network coding header (uncoded) in the packet to be transmitted. This header specifies the packet identifiers (similar to that used by [49, 48]) for the individual packets to be mixed together. Additionally the header carries for each packet an offset giving the offset from the start the packet for the part which has been used for coding, and a boolean specifying if it is the last part. Further, for each packet the location offset (position of start of the part of the packet

in the data block being coded) of the packet in the coded information block will be specified. For example, in Fig. 5.10 the current coding window will pick up the last part from the packet $P_{i+2}(\mathscr{S}_1)$ and starting at some given offset. On the other hand for packet $P_{i+3}(\mathscr{S}_1)$ the offset will be zero and it is not the last fragment, and it will have a location offset equal to one more than the bits used by the part of the packet $P_{i+2}(\mathscr{S}_1)$. This information permits the nodes decoding the data to place it also on a logical tape and then obtain individual packets.

For the case where one of the two streams (logical tapes corresponding to these streams) does not have sufficient data to code with the other stream then the empty space at the end of the current window will be padded for that logical tape with a special *null* packet (contains all zero bits).

However, in order that the sink nodes be able to correctly decode the coded packets, the relay must ensure that it uses the correct packets for coding (i.e. it places the correct packets from the stream end-to-end on the logical tape for the stream). For the case of network coding constellations without overhearing this is not an issue, as the sink nodes will themselves be sources for the packets which they require in order to decode the coded packet and obtain the packets intended for themselves. For network coding constellations requiring overhearing the relay needs to be more careful about the choice of packets it uses for coding.

To understand this aspect let us look again at the example in Fig. 5.9. Let us assume that the node n_{p_1} has reserved the ranges of slots [0, 10] and [15, 20] for the transmissions on the link (n_{p_1}, n_r). From these slots during the course of the SONC initialization handshake the relay has identified the slots [5, 10] as being suitable for overhearing at node n_{s_2}. Further assume that, based on the current demand (corresponding to *resv*) for network coding the relay has notified the nodes n_{p_1} and n_{s_2} that the slots [9, 10] are to be used for overhearing. This implies that the node n_{s_2} will issue pseudogrants for the slots [9, 10] and store data it overhears from the source node n_{p_1} so that it can be used for decoding later. In case the node n_{s_2} would have any further neighbours then the pseudogrants ensure that the node n_{s_2} will be able to overhear transmissions from node n_{p_1} in slots [9, 10] and will not be blocked by any of its other neighbours scheduling transmissions in these slots. Packets which the node n_{p_1} transmits to the relay in the remaining slots reserved for the link between the node n_{p_1} and n_r are not guaranteed to be successfully received and hence available at node n_{s_2}. Hence, the relay should only put the packets belonging to the stream $\mathscr{S}_1$ which have been transmitted in the slots [9, 10] on the logical network coding tape of data corresponding to stream $\mathscr{S}_1$. Here, in order for this to work correctly care has also to be taken by the source node when transmitting the uncoded packets to the relay. This will be discussed later.

Operation at the Sink Nodes

The operation at the sink nodes is simpler than that at the relay. As discussed above the sink nodes will put the coded data received end-to-end in a logical tape. Then using the current window position and the tape of uncoded antidote data packets (only part of this tape is needed and can be identified from the header information discussed above) the sink node will get the packets intended for itself (similar to the explanation for the example in Fig. 2.2).

Which packets the sink node needs to store is known to the sink node at the time of activation of the SONC session (i.e. either overheard packets, or those sent by itself).

Operation at the Source Nodes

The source nodes send uncoded data to the relay. They may need to store this data for decoding purposes if they are also a sink for the network coding constellation. For sources whose transmissions do not need to be overheard by any sink node there is no difference to the normal unicast transmissions. Let us return to the running example which we looked at when discussing the operations at the relay.

In the example let us assume that slots [21, 25] have been reserved for transmissions on the link (n_{s_1}, n_r). The node n_{s_1} may then use any of these slots for transmitting packets belonging to the stream $\mathscr{S}_2$, without any further restrictions. Thus this is similar to the case of normal unicast transmissions.

However, more care needs to be taken by source nodes whose transmissions need to be overheard. Consider the source node n_{p_1}, as discussed in our running example it has the slots [0, 10] and [15, 20] reserved for the unicast transmissions on the link (n_{p_1}, n_r). However, only the slots [9, 10] are suitable for overhearing at node n_{s_2}. Hence, node n_{p_1} will use these slots ([9, 10]) with priority for transmitting the packets belonging to stream $\mathscr{S}_1$. However, remaining packets from the stream $\mathscr{S}_1$ can also be transmitted in the remaining slots if the slots are available for transmission and no other data is pending in the output queue. These packets from stream $\mathscr{S}_1$ cannot, however, be used for coding by the node n_r. A special case arises if only a part (fragmentation is permitted normally at the MAC layer, see [42] for details) of the packet is transmitted in the slots suitable for overhearing. This packet will be needed in a complete form at the sink node which needs to overhear the transmissions. Hence, the remaining fragments of the packet should also be transmitted in slots chosen for overhearing. Hence, pending fragments of such packets which have been partially transmitted in slots suitable for overhearing will be put in a overhearing bucket corresponding to that range of slots. Such remaining fragments will then be transmitted in the slots chosen for overhearing before further packets from the output queue are scheduled for transmission in these slots.

We have thus seen how SONC operations are performed for an active network coding session at the relay node, the source nodes and at the sink nodes. Details about the code for performing the above operation can be found in [115, 116].

5.3.5 Network Coding Session Teardown

During the lifetime of the network coding session, the session master, i.e. the relay node constantly monitors the streams belonging to the network coding session to check if they still satisfy the conditions which enabled the relay to activate the session in the first place. Here the relay keeps track of both the data arrival rates of the individual streams as well as the respective bandwidth reservations. If the necessary conditions for activation are violated for a network coding session then this session is first marked for teardown. Once a decision has been made by the relay to tear down a network coding session, it issues a SONC setup message with type RESET which specifies the network coding session identifier for which it is intended. This message is sent in the data subframe to all the participants of the network coding session (i.e. both the source and the sink nodes). On receipt of this message these nodes are permitted to free the data structures they maintain internally for the network coding session. Grant cancellations will be issued by the individual sink nodes in their respective control subframes for the bandwidth reserved for the multicast reservation. Additionally if bandwidth is being blocked via pseudograntsthen pseudogrant cancellations are also issued by the respective sink nodes (needed only where overhearing was present). In response to the grant cancellations the relay receives from

the sink nodes for its multicast coded transmission it confirms these with a single grant cancel confirmation in the control subframe. The relay sends this confirmation after it has collected the corresponding grant cancels from all the sink nodes in the network coding session.

If conditions in future are favourable for the network coding constellation again then the relay node may activate a new session for the network coding constellation again.

6 Evaluation

The aim of this evaluation is to present a proof-of-concept for SONC. In order to provide this we study representative network coding constellations (given by the respective topology and the flows chosen) and look at the benefits which SONC can provide in each of these representative settings. In Sec. 6.1 we first present an overview of the experimental setups followed by an analysis of selected results in Sec. 6.2.

6.1 Experimental Setups

We performed a thorough performance evaluation of the proposed system. In particular, we implemented all algorithms in our simulation environment that provides a standard-compliant implementation of the optional MeSH mode extensions to the IEEE 802.16-2004 standard which was also developed by us within the scope of this work. Goal of the evaluation is to i) demonstrate the feasibility of the developed concepts for SONC; ii) evaluate the quality of the distributed detection of network coding opportunities; iii) determine the performance (e.g. the detection and activation time for network coding constellations) of the system in realistic settings; iv) determine the performance gains of employing network coding; and v) to gain insights into the overall system behaviour. We perform the following experiments:

- In Exp. (1) we perform an in-depth study of the feasibility of SONC for a basic network coding constellation. We analyze the effectiveness of the distributed detection mechanism and illustrate the resulting changes to the bandwidth reservations.
- Exp. (2) presents a more challenging traffic constellation with overlapping antiparallel flows and more degrees of freedom w.r.t. the choice of appropriate relay nodes. We focus on the responsiveness and timing of the system in our discussion.
- In Exp. (3) we study a diverging flow constellation in a split topology. Here the challenge is to realize that the flows can be coded, despite the fact that they are splitting. We hereby also study the implications of SONC with overhearing.
- Exp. (4) serves to investigate the trade-offs involved if we used SONC with overhearing in topology with a highly connected sink node which needs to overheard uncoded transmissions from a source node, and diverging flows.
- Exp. (5) also studies SONC with overhearing. In contrast to Exp. (4), we investigate a topology with a highly connected relay node.
- In Exp. (6) we investigate a butterfly topology with cross flows. Similar to Exp. (3), the challenge is to detect the possibility to code the streams given by these flows.
- We study the potential NC gain for all experiments using the degree of freedom (see Chap. A) metric.

The common parameters for all simulations are detailed in Table B.2, the basic PHY layer parameters employed in our simulations are given in Table B.1 both of which can be found in Sec. B.4.

If not noted otherwise, we perform 20 replications for each experiment. For the timing parameters, we give the average values for all experiments (see Table 6.5). For the performance gain, we utilize the degree of freedom metric introduced earlier. In particular, we give the average $NormCurrTotDof^t$ (see Chapter A) over all nodes in the network (see Fig. 6.10) for the

simulation duration. To simplify the presentation in the tables and figures this is shortened as NDoF or Network-DoF. Further, as we are using SONC as the network coding mechanism we shorten it to NC and use the two terms interchangeably in this chapter where this is clear from the context.

6.2 Analysis of Results

Exp. (1)

Exp. (1) was used to study the feasibility of the distributed network coding constellation detection mechanisms. Moreover, we obtain the timing parameters (given in frame numbers, each frame of length 20ms) of the overall system, which illustrate the efficiency of the individual aspects of SONC. We study the basic line topology shown in Fig. 6.1 and utilize the flow setups given in Table 6.1.

Table 6.1: Flow Configuration for Exp. (1)

Flow	Nodes	Rate [bit/frame]	Start → Stop Frame
f_1	3 → 1	18000	300 → 2700
f_2	1 → 3	18000	0 → 2500

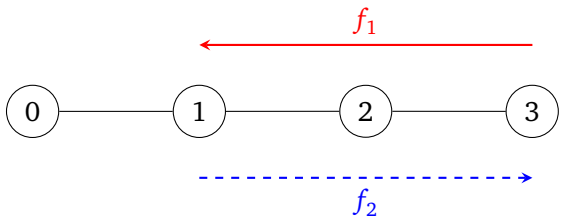


Figure 6.1: Line Topology with Antiparallel Flows - Exp. (1)

The results clearly indicate that the developed systems operates according to the specification. The timing results (see Table 6.5) indicate the proper working of the proposed distributed mechanisms to detect streams, the protocols to reserve multicast bandwidth and the NC operation itself. For the given setting NC can be beneficially deployed, since it yields a significant performance gain as shown in Fig. 6.10(a) using the NDoF metric.

To illustrate the operation of NC in combination with the studied reservation-based MAC, for Exp. (1), we plot the reserved number of minislots for the unicast and multicast links established by relay node 2. See Fig. 6.2 for the setup without NC and Fig. 6.3 for the setup with NC being active.

It is clearly visible that our mechanisms replace the unicast reservations (see Fig. 6.2) using multicast reservations (see Fig. 6.3), thus increasing the efficiency of the system. The initial overshooting of the unicast reservations is due to the transfer of the packets that have been queued while the multihop route is still being established.

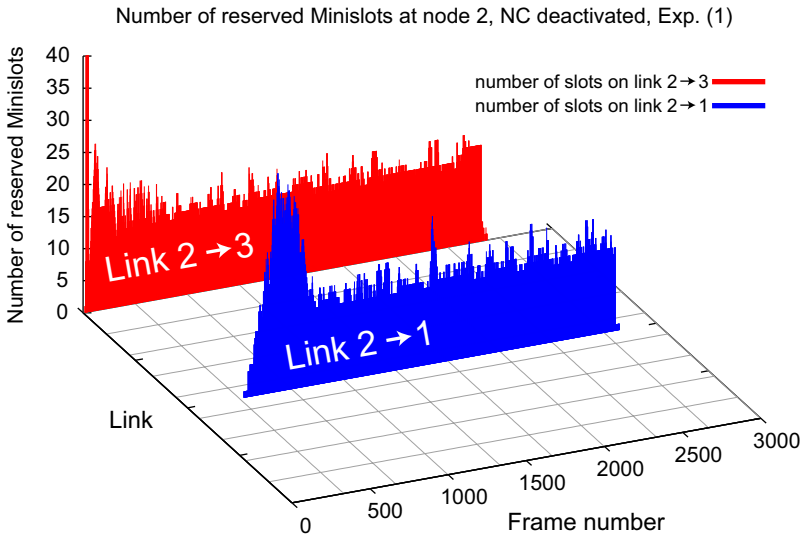


Figure 6.2: Number of Reserved Minislots at Relay Node 2 for Exp. (1) with NC Being Inactive

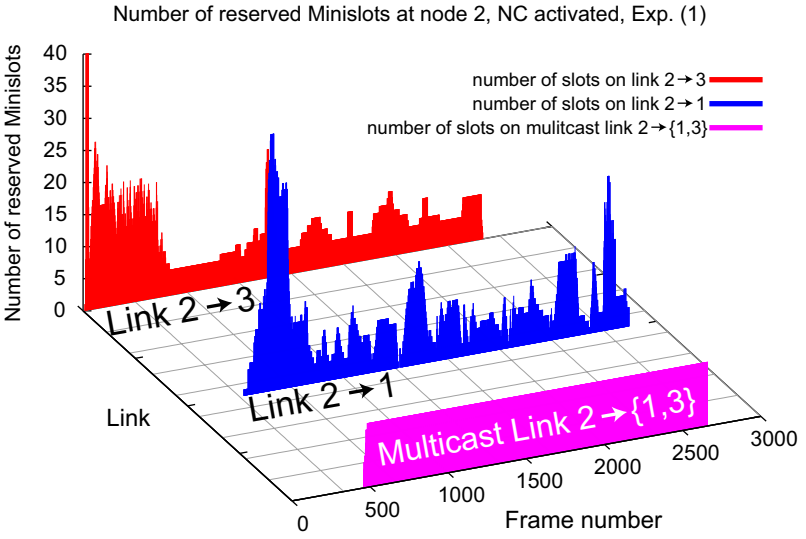


Figure 6.3: Number of Reserved Minislots at Relay Node 2 for Exp. (1) with NC Being Active

In Fig. 6.3 it is clearly visible that multicast reservation covers a constant bit rate, i.e., it accounts for the part of the flows (streams composed from the flows*) that have been identified within the flow (stream) set as suitable for NC using our distributed mechanisms (see Sec. 5.3 for details). The remaining unicast reservations account for the part of the flows that cannot be coded as well as for control messages.

Exp. (2)

Table 6.2 shows the flow setup for Exp. (2), the employed topology is shown in Fig. 6.4. We focus on the functional analysis of the system using the obtained timing parameters, which are given in Table 6.5 for the different relays available in this topology (node 2, node 3 and node 4). The obtained performance gain for the network is shown in Fig. 6.10(b).

Table 6.2: Flow Configuration for Exp. (2)

Flow	Nodes	Rate [bit/frame]	Start → Stop Frame
f_1	4 → 1	3600	200 → 2400
f_2	1 → 4	7200	0 → 1500
f_3	2 → 5	3600	1000 → 2700
f_4	5 → 2	6000	2000 → 2800

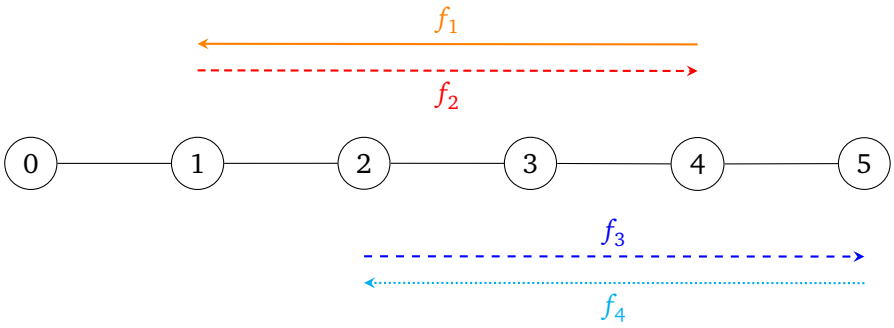


Figure 6.4: Line Topology with Parallel Pairs of Antiparallel Flows - Exp. (2)

Given the topology for Exp. (2) (see Fig. 6.4), and the flows we have only three nodes which can act as relay nodes for network coding constellations. These nodes are node 2, node 3, and node 4. Node 2 can be a relay for network coding constellation $\mathfrak{N}^2=(2, \{\mathcal{S}_2^{1,3}, \mathcal{S}_2^{3,1}\})$, similarly nodes 3 and 4 can be relays for the network coding constellations $\mathfrak{N}^3=(3, \{\mathcal{S}_3^{2,4}, \mathcal{S}_3^{4,2}\})$ and $\mathfrak{N}^4=(4, \{\mathcal{S}_4^{3,5}, \mathcal{S}_4^{5,3}\})$ respectively.

From the flow start and stop times given in Tab. 6.2 we can derive the following theoretical ideal lifetimes for the individual streams (specified as range of frames where the stream exists): $\mathcal{S}_2^{1,3}$: [0, 1500], $\mathcal{S}_2^{3,1}$: [200, 2400], $\mathcal{S}_3^{2,4}$: [0, 2700], $\mathcal{S}_3^{4,2}$: [200, 2800], $\mathcal{S}_4^{3,5}$: [1000, 2700], and $\mathcal{S}_4^{5,3}$: [2000, 2800].

Using Exp. (2) we exemplify the performance of our solution to detect network coding constellations and identify network coding opportunities and then initiate network coding sessions.

* Where a stream consists of packets from exactly a single flow we use the terms flow and stream interchangeably where the meaning is clear from the context of the discussion.

We demonstrate the performance using the obtained timing parameters (given in frame numbers, each frame of length 20ms). Our timing results for relay node 2 are as follows. $T_{\text{start}} = 200$ is the earliest potential point in time for starting SONC (only theoretically feasible, though) at relay node 2. Here, starting at this frame we have the two flows f_1 and f_2 in the network and so at node 2 two streams can be detected (i.e. the network coding constellation $\mathfrak{N}^2$ can be detected). In average over all experiments, the network coding constellation given by flow set (f_1 and f_2 and the corresponding streams) was detected at frame $T_{\text{FSD}} = 236$; followed by the possibility to employ NC at $T_{\text{NCF}} = 364$. The SONC initialization handshake was finished with receiving the ACTIVATE message at $T_{\text{ACTIVATE}} = 365$ (i.e. a delay of 3.3s vs. the earliest theoretical point of time for coding) while the earliest successfully decoded packet was registered at $T_{\text{DEC}} = 479$. One of the flows in the flow set ceased to exist at $T_{\text{stop}} = 1500$, which in average triggered a timeout for the network coding session at $T_{\text{NCTO}} = 1645$ (i.e. 2.9s after the theoretical earliest possible point in time for detection), while the change in the set of stream has again been discovered at $T_{\text{CHG}} = 1527$ with a very short delay. We can compare this with the ideal duration for existence of the network coding constellation $\mathfrak{N}^2$, the frames [200, 1500]. The obtained timing values clearly show that our system is able to swiftly detect streams, find network coding constellations and obtain the possible NC opportunities.

Similar results can be seen at the other relay nodes. Of special interest is the case of relay 3. Here, the streams $\mathscr{S}_3^{2,4}$ and $\mathscr{S}_3^{4,2}$ are each composed of packets belonging to multiple flows which are not present for all the time in parallel. Here, the timing parameters for relay node 3 in Tab. 6.5 show that the duration over which the relay detects the presence of the network coding constellation $\mathfrak{N}^3$ [335, 2848]; is very close to the range ideally possible in theory [200, 2700] and that the mechanisms are not disturbed by the leaving of the older flow and joining of newer flows for a stream. In fact the individual sets of streams are identified even earlier (frame 207) and it takes up to frame 335 for the relay to deem the network coding constellation as a meaningful constellation which can be activated (see discussion in Sec. 5.3.2).

Exp. (3)

The setup of Exp. (3) is more challenging for the detection of the coding opportunities due to the split flow setting. Also, the possibility of overhearing exists. Table 6.3 shows the corresponding flow setup (also used for Exp. (4) and Exp. (5)), the topology is shown in Fig. 6.5.

Table 6.3: Flow Config. for Exp. (3), (4) and (5)

Flow	Nodes	Rate [bit/frame]	Start → Stop Frame
f_1	$1 \rightarrow 4$	5000	$0 \rightarrow 2000$
f_2	$4 \rightarrow 2$	3000	$300 \rightarrow 2300$

Our results show that the system is able to reliably detect the NC constellation. Table 6.5 gives the timing parameters for relay node 3. We observe that NC is not immediately started after having discovered the NC opportunity (T_{NCD}), because at this time the system still lacks the necessary bandwidth reservation to enable overhearing of node 2. Our system automatically reserves the bandwidth and at T_{NCF} all requirements for overhearing are fulfilled. The NC session is established shortly afterwards and results in performance gains (see Fig. 6.10(c)).

In Fig. 6.6 we exemplify the measured timing parameters in the context of the simulated flows for Exp. (3), which gives a better visual representation of the operation of the system. Please refer to Table 6.5 for a definition of the timing parameters shown.

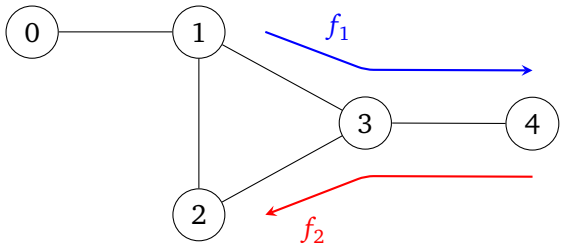


Figure 6.5: Split Topology with Diverging Flows - Exp. (3)

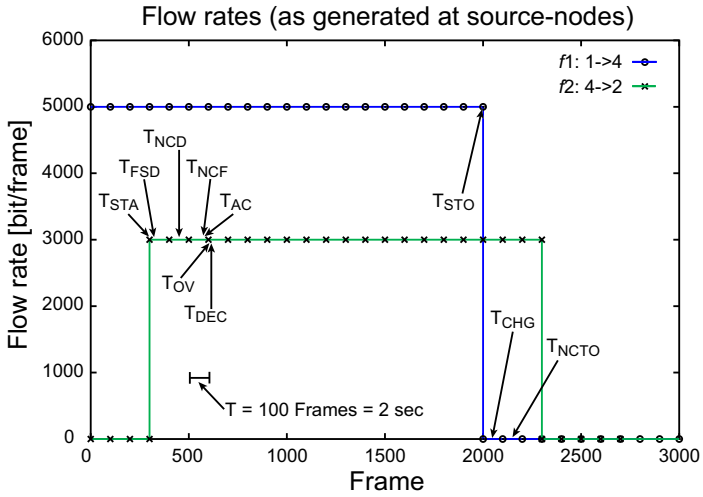


Figure 6.6: Flow Setup and Measured Timing Parameters for Exp. (3)

Exp. (4)

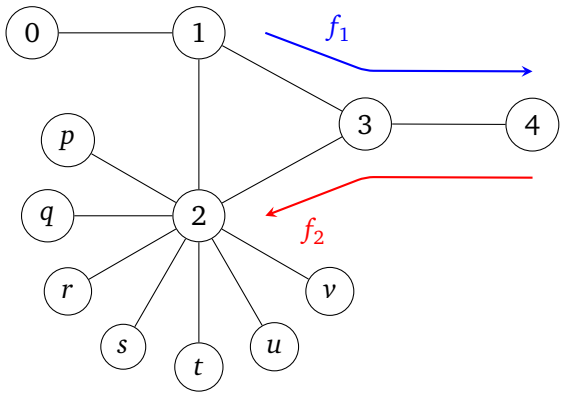


Figure 6.7: Split Topology with Diverging Flows - Exp. (4)

Exp. (4) facilitates a more detailed study of the trade-offs involved if using overhearing. The topology has been changed from Exp. (3) by adding a number of (inactive) neighbours to the receiving node 2 (see Table 6.3 for the corresponding flow setup, the topology is shown in Fig. 6.7). As a result, reserving bandwidth to enable overhearing blocks availabilities at these receivers, which directly, negatively influences the degree of freedom of the network. As a result, activation of NC in this topology results in a reduced NDoF, thus indicating a performance degradation if using NC as shown in Fig. 6.10(d). Hence, we can conclude that it is necessary to

carefully study the topology and flow configuration before activating NC, since NC is no panacea and does not always yield performance gains. Based on the DOF metric, we can easily check, whether we expect gain from deploying NC; we have implemented this check in our system and verified using experiments that indicate its reliable operation. The timing parameters obtained differ slightly from the ones obtained in Exp. (3) (see Table 6.5), because the system has been able to reserve the overhearing bandwidth instantly.

Exp. (5)

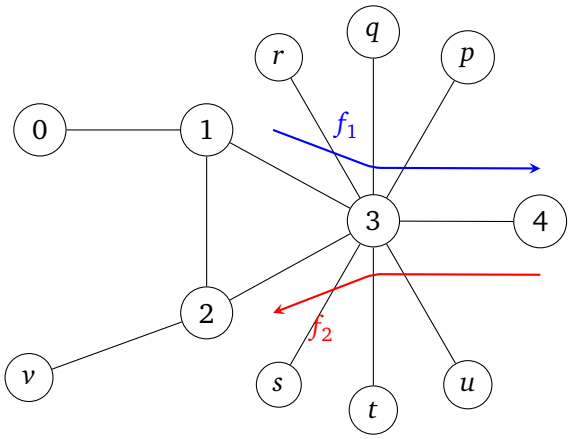


Figure 6.8: Split Topology with Diverging Flows - Exp. (5)

In contrast to Exp. (4), Exp. (5) illustrates an example, where overhearing is in fact beneficial in a quite highly connected topology. See Table 6.3 for the corresponding flow setup, the topology is shown in Fig. 6.8. Here, the relay node 3 has a number of (inactive) neighbours. These neighbours are not affected by enabling the overhearing at node 2 and, hence, do not cause an additional loss in DoF for the network. The gain obtained using NC is shown in Fig. 6.10(e). The timing parameters obtained are quite similar to Exp. (4) and are given in Table 6.5.

Exp. (6)

Our final test in Exp. (6) serves to show the feasibility of NC for within a butterfly topology with cross flows. See Table 6.4 for the corresponding flow setup, the topology is shown in Fig. 6.9. The given setting is challenging, because the relay has to detect the cross flows in combination with two potential overhearing possibilities.

Table 6.4: Flow Configuration for Exp. (6)

Flow	Nodes	Rate [bit/frame]	Start → Stop Frame
f_1	1 → 5	5000	0 → 2000
f_2	4 → 2	3000	300 → 2300

Again, Table 6.5 gives the relevant timing parameters obtained for the relay node 3. We observe that the NC opportunities are detected reliably and fast. However, the activation of NC is deferred, which can be explained due to the more challenging overhearing constellation. In the

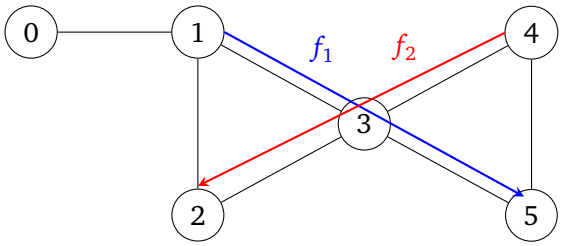


Figure 6.9: Butterfly Topology with Cross Flows - Exp. (6)

given example, a number of long-term reservations have to be performed to enable overhearing of both overhearing links (for nodes 2 and 5), which accounts for the delay. Again, the activation of flow-based NC results in a significant performance gain as shown in Fig. 6.10(f).

6.3 Summary of Results

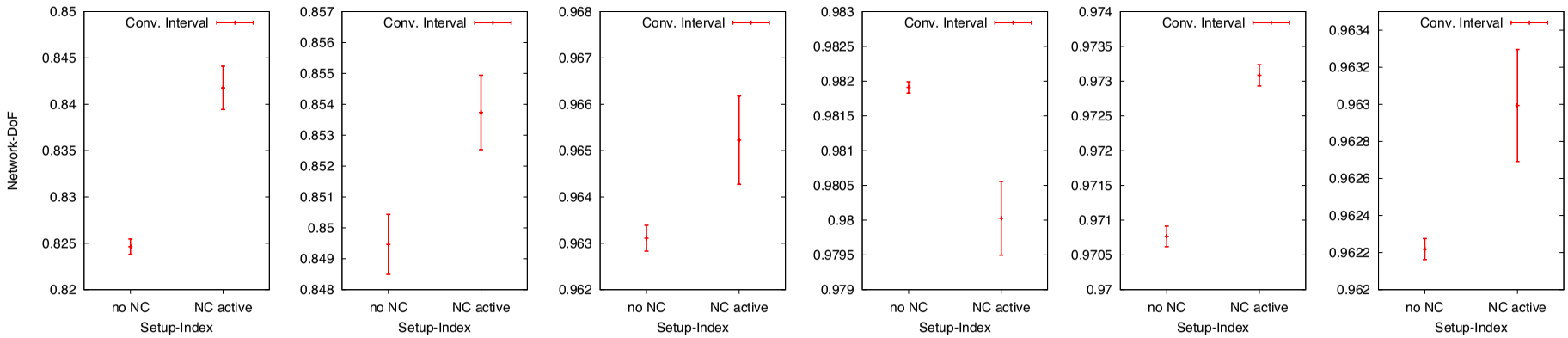
In summary, the obtained values indicate that the performance of the developed scheme is very good. NC opportunities are detected reliably and used nearly immediately after the possibilities arise, even taking into account the time-consuming handshake process to reserve bandwidth in the investigated IEEE 802.16 network. Using our NDoF metric, the relay is also able to quickly check if NC is beneficial in the given setting; thus enabling an informed decision whether to code or not to code. The entire system has been shown to operate efficiently and according to the specifications.

We have thus validated the functional properties (detection of network coding constellations, selection of network coding constellations for activation, efficient setup of network coding sessions, operation of SONC, and teardown of network coding sessions) of SONC and provided a proof-of-concept for SONC.

Table 6.5: Summary of Experimentally Obtained Timing and Performance Parameters

Exp.	Relay	T_{STA}	T_{FSD}	T_{NCD}	T_{NCF}	T_{AC}	T_{OV}	T_{DEC}	T_{STO}	T_{CHG}	T_{NCTO}	$NDoF_{noNC}$	$NDoF_{NC}$
(1)	2	300	327	n.a.	455	456	n.a.	467	2500	2540	2652	0.8246±8e-04	0.8417±2e-03
(2)	2	200	236	n.a.	364	365	n.a.	479	1500	1527	1645	0.8495±9e-04	0.8537±1e-03
(2)	3	200	207	n.a.	335	337	n.a.	352	2700	2725	2848	0.8495±9e-04	0.8537±1e-03
(2)	4	2000	2009	n.a.	2132	2134	n.a.	2144	2700	2724	2853	0.8495±9e-04	0.8537±1e-03
(3)	3	300	322	450	578	586	603	616	2000	2051	2153	0.9631±3e-04	0.9652±1e-03
(4)	3	300	313	n.a.	438	449	454	468	2000	2052	2154	0.9819±8e-05	0.9800±5e-04
(5)	3	300	316	n.a.	444	454	460	474	2000	2044	2154	0.9708±2e-04	0.9731±2e-04
(6)	3	300	324	449	1131	1138	1141	1149	2000	2062	2153	0.9622±6e-05	0.9630±3e-04

Measured timing variables. T_{STA}: earliest starting time for NC (only theoretical possibility); T_{FSD}: detection of first stream set; T_{NCD}: detection of potential NC session, but NC not applicable due to lacking bandwidth reservation for overhearing (only for some scenarios with overhearing); T_{NCF}: detection of feasible NC session; T_{AC}: completed NC INIT handshake, successfully received ACTIVATE message; T_{OV}: earliest overhearing of coded data by node 2 (only for overhearing scenarios); T_{DEC}: earliest successful decoding of data; T_{STO}: stopping time for one flow of the flow set; T_{CHG}: detection of the flow change; T_{NCTO}: firing of the time out for a outdated network coding session (i.e. streams/flows have ceased to exist).
Measured performance variables. $NDoF_{noNC}$: average network degree-of-freedom without NC; $NDoF_{NC}$: average network degree-of-freedom with NC.



(a) Exp. (1) (b) Exp. (2) (c) Exp. (3) (d) Exp. (4) (e) Exp. (5) (f) Exp. (6)

Figure 6.10: Network Degree-of-Freedom for Selected Experiments with 95% Confidence Intervals

Part III

Global Mechanisms for Efficient Deployment of Network Coding

7 CORE Framework Overview

7.1 Motivation for the CORE Framework

In the former part of the thesis we presented efficient mechanisms for enabling network coding to be deployed in WMNs using bandwidth reservation for providing QoS. These mechanisms allow bandwidth savings via locally analyzing streams of packets which cross an individual node. However, it is also vital that the routing in the WMNs routes the packets such that network coding can be applied where possible. We will motivate this with a simple example.

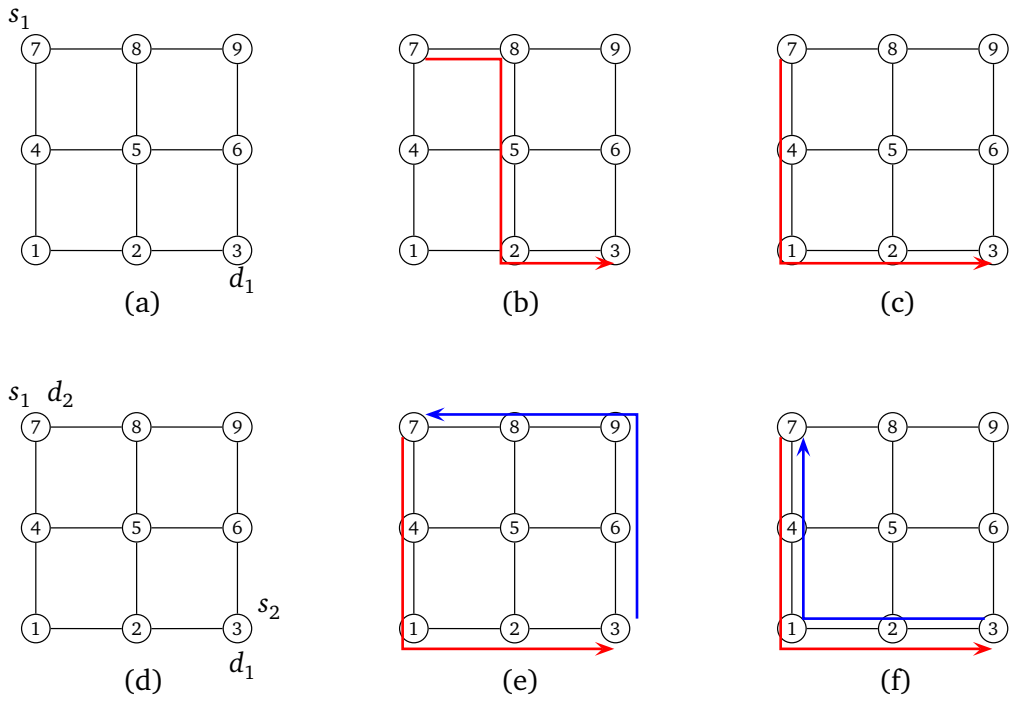


Figure 7.1: Simple Example Explaining the Motivation for CORE

Consider Fig. 7.1 (a). Assume that we have the wireless mesh network in the form of the grid topology as shown with a single flow $f(s_1, d_1)$ in the network. Assume that the routing used in the WMN is shortest hop routing*.

Fig. 7.1 (b) shows a possible shortest path which the packets for the flow can take. This is a best path, given the routing metric (number of hops), however, is bad for the WMN on a whole. This is due to the fact that the packets for the flow $f(s_1, d_1)$ are routed via very densely connected nodes in the network (e.g. node 5). Due to this the opportunities for spatial reuse of bandwidth are reduced in the WMN. Our aim in this thesis, is however to improve the throughput in the

* Shortest hop routing combined with sufficient bandwidth reservation along the path will under normal conditions give the best QoS performance in terms of delay for individual flows in the WMN.

WMN (also by using network coding). Hence, the shortest path shown in Fig. 7.1 (b), although an optimal choice from point of view of the individual flow, is a suboptimal choice for the WMN on a whole. We may instead route the same flow $f(s_1, d_1)$, as shown in Fig. 7.1 (c). This route is also a shortest hop path, and hence is also an optimal route from the point of view of the flow $f(s_1, d_1)$. However, the route in Fig. 7.1 (c), avoids the densely connected node in the center of the grid topology. This opens up more opportunities for spatial reuse in the WMN. Thus, we can increase the traffic carrying capacity of the network by avoiding such densely connected nodes in the WMN where possible. This approach is also suggested by other work found in the literature (see for e.g. Ref. [114] and the discussion in Chap. 2).

Let us now assume, we use such an enhanced interference-aware shortest path routing approach (as seen in Fig. 7.1 (c)). We will now look at the same WMN topology as earlier but consider that this time we have two flows in the network as shown in Fig. 7.1 (d). These two flows are: flow $f(s_1, d_1)$, with the earlier source and destinations, and the flow $f(s_2, d_2)$, with the source and destination as shown in Fig. 7.1 (d). Now, we can use the interference-aware shortest path routing for each flow to obtain, for example, the routes as shown in Fig. 7.1 (e). Here, each flow is using a shortest hop path, which is furthermore interference-aware, in a sense that it avoids the densely connected nodes in the WMN thereby reducing the interference, and hence blocked links in the WMN when individual packets are transmitted on the paths. However, the given choice of routes in Fig. 7.1 (e) permits no additional bandwidth savings via network coding. Thus, if the routing considers only QoS (shortest path), and the interference (number of links blocked), it is not guaranteed that network coding will be able to save further transmission bandwidth in the WMN. We could on the other hand route these two flows using the routes shown in Fig. 7.1 (f). Here, we have the shortest hop path for each flow, additionally, these routes lead to the least number of blocked links, and are thus interference-aware. But, unlike in Fig. 7.1 (e), using the routes in Fig. 7.1 (f) it is possible to deploy network coding in the network (e.g. at nodes 1, 2, and 4) thereby achieving additional bandwidth savings.

Thus, to make effective use of network coding in the network the routing has to be aware of the possibility of network coding and actively route flows to generate network coding opportunities. However, as shown in our simple example, we should also take care to choose paths which avoid blocking a huge number of nodes (links) in the WMN to enhance the potential for spatial reuse of bandwidth. Additionally, when choosing the routes, the routing used should also consider if the chosen routes are feasible from the scheduling point of view. This is especially important in reservation based WMNs where QoS in terms of delays is very vital to the applications in the WMN. If there is not sufficient bandwidth available for the aggregate traffic along the paths chosen for the individual flows, it will lead to huge queueing delays for packets at the individual nodes in the WMN. Thus, routing should also be schedule aware in a sense that the routes chosen are feasible from the point of view of the bandwidth requirements of the flows.

This essentially leads us to conclude (similar to the work [104]) that network coding, routing, and scheduling have to be jointly optimized in order to achieve the maximum benefit from network coding in the WMN.

We next present our solution termed CORE (for Centrally Optimized Routing Extension) which jointly optimizes routing, scheduling, and network coding in the WMN. CORE addresses the issues which we highlighted using the simple motivating example. The key properties of CORE can be listed as follows:

- CORE is able to work in WMNs using standard routing protocols. It only adapts the routes for individual flows by adapting the routing tables at the individual nodes. CORE adapts the routes to approach its optimization goal, i.e. jointly optimize the QoS aware routing, transmission schedule and the bandwidth savings via application of network coding in the WMN.

- CORE uses heuristics run centrally (run at a given node, termed as the central server) to compute and plan the routes and route adaptations. CORE is designed such that the network operator can parametrize CORE's heuristics to limit computational costs to a given maximum threshold. Thus, the network provider can configure CORE such that the computation of the solutions and their deployment in the WMN is possible in near real-time. Furthermore, this enables CORE to be deployed in WMNs with realistic scenarios and dynamically changing traffic demands.
- CORE's design uses distributed components (routing, distributed scheduling) to deploy the solutions optimized centrally. This reduces the WMNs dependence on central control, and permits normal operation of the WMN even if the central server running CORE's heuristic fails. Another benefit of our design decision is that CORE's central server does not need to maintain complete global information about the nodes in the WMN. E.g. the central server does not need to know the individual transmission schedules at the individual nodes in the WMN. This not only reduces the complexity of the solution, but also reduces the amount of information needed to be gathered and maintained up-to-date centrally. Which means that CORE does not involve significant control and traffic overhead in the WMN.
- CORE's functioning is tailor made to enable efficient operation of stream-oriented network coding.

7.2 CORE: Logical Workflow

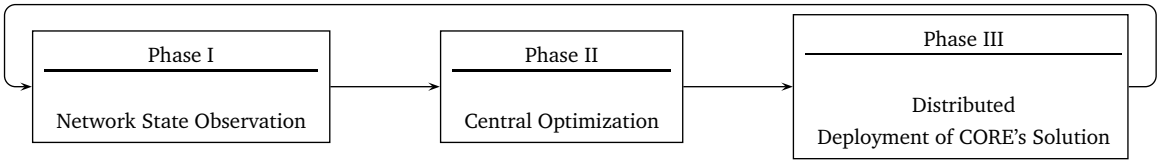


Figure 7.2: Logical Workflow of CORE

We will now present a brief overview of the logical flow of operation of CORE. Consider Fig. 7.2; as shown in the figure CORE's working can be divided into three logical phases, where Phase I precedes Phase II and so on. However, it does not mean that during working these three operational phases run strictly sequentially. In fact it is likely that all three phases are concurrently active and in progress in the network.

We will now shortly identify and list the functional tasks of each phase:

- Network State Observation: Here, the individual node in the network observes the traffic flows originating (entering the WMN) at itself. Each node maintains statistics about the data arrival rate for each such flow in order to estimate the long-term[†] data arrival rates for each flow. Whenever new long-term flows are identified or the traffic demands for individual flows change beyond a selected threshold, update messages are sent by the nodes to the CORE central server. Further periodic updates are sent to the central server for flows for which no updates have been sent in the recent past. Additionally, the CORE central server is responsible for maintaining a set of routes which are feasible from the point of view of each flow considering the given routing metric. We assume here that the server tries to route flows along the shortest path (given the routing metric), however,

[†] Here long-term flow means flows with steady data arrival rate of over a chosen number of frames. This is important to avoid optimizations for transient bursts of traffic.

is also free to choose slightly longer (within a given threshold from the shortest path) paths for individual flows in order to optimize the network as a whole. The central server observes the currently reported set of flows in the network and maintains a suitable set of alternate paths (besides the shortest path) for each long-term flow in the network.

- Central Optimization: Using the information obtained about the long-term flows in the network, and the preselected set of alternative routes for each flow, CORE uses its heuristics to compute feasible route combinations (one route per flow). These routes are selected such that if the packets are routed along the computed routes it should be possible to obtain a feasible transmission schedule where each node in the network is able to reserve sufficient bandwidth for the flows in the network.
- Distributed Deployment of CORE's solution: After computing the route combination and the transmission schedule, the CORE central server sends control messages to the affected nodes in the network notifying them of the solution computed by it. As a response to the received control messages individual nodes adapt their routes and transmission schedules in a coordinated manner. The individual nodes in the WMN are responsible for changing their local routing tables as indicated by CORE's control messages. Further, the nodes are also responsible for reserving bandwidth for their data transmissions, and for setting up network coding sessions and reserving bandwidth for their network coding sessions. Thus, CORE; although it computes the optimized route combination centrally at the server; uses distributed mechanisms for its deployment.

In this chapter we motivated the need for the CORE framework. We identified the key properties of CORE and presented a simplified overview of its logical functioning. In the next chapter we will provide more details about the optimization problem CORE seeks to solve and also provide details for the individual components of the CORE framework.

8 CORE Framework Details

In this chapter we discuss the details of the CORE Framework. CORE aims to tackle the joint routing, scheduling, and network coding problem we discussed earlier in Sec. 7.1. We shall first (in Sec. 8.1) present more details and a concrete description of the optimization problem CORE aims to solve. We thereby also present the details of the network model and notation we need for the description of the optimization problem; and those notations which will also be used later in the thesis. This is followed by details of the heuristics used by CORE to achieve its optimization goal in WMNs in Sec. 8.4.

8.1 CORE's Optimization Problem and Notation

In this section we present the optimization goal of CORE. We first introduce some formal notation which will be used in this chapter and the thesis, as well as present an introduction to selected metrics and definitions which are used later in this thesis.

8.1.1 Network Model and Definitions

Consider that the wireless mesh network consists of a set of N nodes. The WMN is modelled as a graph $G=(\mathscr{V},\mathscr{E})$ with the set of vertices $\mathscr{V}$ representing the N nodes in the network, and the edge set $\mathscr{E}$ representing the set of E wireless links in the network. A wireless link (i,j), where $i, j \in \{1, 2, \ldots, N\}$, exists in the network if it is possible for node j to correctly receive the transmissions from node i in the absence of any other simultaneous transmissions in the WMN.

We denote the transmitter of edge $e \in \mathscr{E}$ by T_e and the receiver of edge e by R_e. Thus, given an edge $e=(i,j) \in \mathscr{E}$, where $i, j \in \mathscr{V}$, $T_e=i$ and $R_e=j$. The graph G is not assumed to be fully connected which means that the nodes communicate with each other via multihop routes. Further, as per the IEEE 802.16 constraints we assume that if an edge $e_1=(i,j) \in \mathscr{E}$ then edge $e_2=(j,i) \in \mathscr{E}$. Given edge $e=(i,j)$ we will denote its reverse edge as $\overline{e}=(j,i)$. The set of outgoing links or edges at a node n will be denoted by $\mathscr{E}_n^{out}$. Thus $\mathscr{E}_n^{out}=\{e\colon T_e=n,\ e \in \mathscr{E}\}$. Similarly, the set of incoming links or edges at a node n will be denoted by $\mathscr{E}_n^{in}$. Thus $\mathscr{E}_n^{in}=\{e\colon R_e=n,\ e \in \mathscr{E}\}$. We define the set of neighbours of a node n as $Nbr(n)=\{n_i\colon (n,n_i) \in \mathscr{E}\}$. The set of links which belong to a route is defined as a path. Thus a path $\mathscr{P}_s^d$ is a ordered list of edges $\subseteq \mathscr{E}$ which form the route between source node s and destination node d. The order of the edges in the path specifies the order in which the links are traversed on the route between the source node (denoted as $S_{\mathscr{P}}$) and the destination of the path (denoted as $D_{\mathscr{P}}$).

We will now look into detail at the scheduling model (introduced in Chap. 3) we have for our wireless mesh network, and define it formally. We define the set of edges which interfere with transmissions of an edge e to be the set $I(e)$ which is given by Eq. (8.1).

$$
\begin{aligned}
I(e) \;=\; & \mathscr{E}^{out}_{T_e} \cup \mathscr{E}^{in}_{T_e} \\
& \cup \mathscr{E}^{out}_{R_e} \cup \mathscr{E}^{in}_{R_e} \\
& \cup \forall_{n_r \in Nbr(R_e)} \mathscr{E}^{out}_{n_r} \\
& \cup \forall_{n_t \in Nbr(T_e)} \mathscr{E}^{in}_{n_t} \\
& - e
\end{aligned} \tag{8.1}
$$

Thus, Eq. 8.1 basically formally states the scheduling constraints specified by the IEEE 802.16 standard, which we use for modelling and planning the schedule in our WMN. The above can be compared to the graph based scheduling and spatial reuse model found in the literature (see for e.g. [55]). The model prevents both primary conflicts (i.e. the transmitter or receiver are carrying out other transmissions or receptions in parallel) as well as secondary conflicts at the receivers. Thus no neighbour of the receiver may be allowed to transmit, and using the same logic, no further neighbour (other than R_e) is permitted to receive when transmissions are scheduled on edge e. Thus, this model permits more aggressive spatial reuse than that permitted by the RTS/CTS mechanism of the IEEE 802.11 standard where secondary transmitter conflicts are also not permitted, i.e. the neighbours of the transmitter may also not transmit. Fig. 8.1 visualizes the concept of $I(e)$.

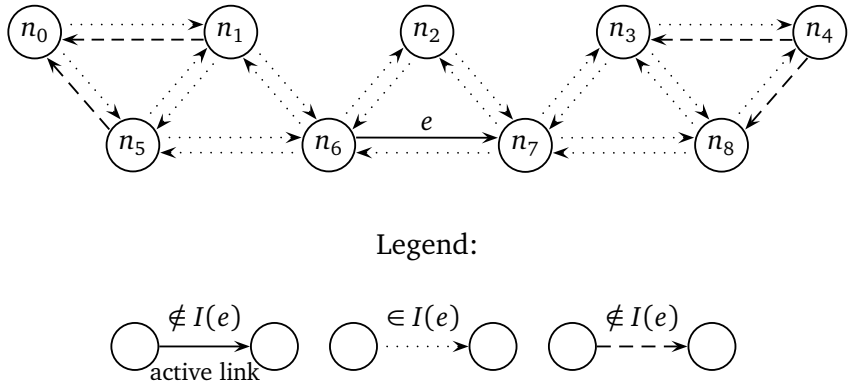


Figure 8.1: Visualization of $I(e)$

As seen from Fig. 8.1 the edge e is not considered to interfere with itself for convenience. The reason for this is that it makes no sense to schedule multiple transmissions on the same link which occur simultaneously. As the transmission capacity of edge e can be used only once. Thus, from Fig. 8.1 one can see that even when a single link in the WMN is activated a number of links (those belonging to the set $I(e)$) are prevented from carrying data transmissions in parallel. We can thus use the above fact to measure and quantify the cost of scheduling transmissions on a given edge e in the WMN.

$$
\varsigma(e) = |I(e)| \tag{8.2}
$$

Thus, as shown in Eq. 8.2, we define the blocking-cost of transmission on an edge e as $\varsigma(e) = |I(e)|$. Which basically counts the number of edges in the graph which are blocked by a transmission scheduled on edge e in the WMN.

Now, analogously, we define the blocking-cost for a path (route) $\mathscr{P}_s^d$ between source s and destination d to be $\varsigma^{sd}\left(\mathscr{P}_s^d\right)$ and is computed as shown in Eq. (8.3).

$$\varsigma^{sd}\left(\mathscr{P}_s^d\right) = \sum_{e_i \in \mathscr{P}_s^d} \left|I(e_i)\right| \tag{8.3}$$

Thus, as seen from Eq. (8.3), the $\varsigma^{sd}\left(\mathscr{P}_s^d\right)$ provides an upper bound on the number of links which will be blocked when all links on the path are transmitting, and hence is an pessimistic estimate, but still serves the purpose of cost estimation. Note that the actual number of links which are blocked when transmissions are scheduled on individual links in a path can be less than the value given by Eq. (8.3). This is as it might be possible for multiple links on a given path to be scheduled to transmit simultaneously, furthermore, a given link if blocked by such transmissions is counted multiple times (once for each edge on the path), although in reality it will be blocked only once. However, our simplification allows us to compute the blocking cost of a path without having full knowledge of the transmission schedule. This simplifies the computational complexity of computing the blocking cost of a path, as well as permits this metric to be used by standard routing algorithms, e.g. Dijkstra's shortest path algorithm (see Ref. [18]), for computing the minimum interference path.

8.1.2 Optimization Problem Definition

Having seen the most important definitions and the mathematical model for our network and scheduling constraints in Sec. 8.1.1 we will now specify the optimization problem which CORE targets. CORE's optimization goal can be specified as given by Eq. (8.4).

$$\text{Maximize: } \sum_{i=0}^{\mathscr{F}} A\left(f_i\right) \cdot V\left(f_i\right) \tag{8.4}$$

Eq. (8.4) basically approaches the problem from a network operator's point of view, which is also the focus of our work. Here, $A(f_i)$ is an indicator function which indicates whether a feasible route is found for the flow identified by the index i. A route is considered to be feasible for the flow, if it is permissible with respect to the required QoS specifications, and in addition a feasible conflict-free schedule can be found which allows the bandwidth demand of the flow to be satisfied; given all the other flows which have already been allocated routes in the network and that their traffic demands are also satisfied. Thus, $A(f_i)$ takes the value 1 if a flow is scheduled and the value 0 if it is not allocated a feasible route in the WMN. $V(f_i)$ is the value or benefit which the operator associates with the flow f_i. Thus, $V(f_i)$ can be seen as the benefit which the network operator would obtain if the flow f_i is admitted into the WMN such that it is allocated a feasible route. For example considering that customers pay per unit of data which is carried by the WMN, the operator would aim to maximize the total volume of data traffic which is carried by the network. In this case $V(f_i)$ may be given by the traffic demand of the flow f_i. As our goal is to increase the traffic carrying capacity of the WMN, in this thesis, we will assume that the value of a flow is directly proportional to its traffic demand. However, the fact that the routes chosen need to be feasible with respect to the QoS permits as well as scheduling constraints means that we need to constrain the above optimization problem. The constraints for the above optimization problem are specified next.

The optimization objective specified in Eq. (8.4) has to be achieved subject to the constraints $C_1 \ldots C_9$ shown in Eq. (8.5) ... Eq. (8.13). Constraint C_1 specifies that the value or benefit which the network provider obtains by admitting a flow in the network is always positive. This is justifiable as when adding a new flow into the network we always see if there are sufficient bandwidth resources available for the new flow without necessarily taking away resources already allocated to other flows. Further, this assumption is also valid in a scenario where the operator profits from each flow admitted to the network and the additional profits by carrying the additional traffic volume the flow introduces into the WMN. Here, the term b_i denotes the traffic demand in minislots per frame needed by flow f_i.

Let $\mathscr{R}_{f_i}$ denote the set of routes which are permissible with respect to the QoS specifications of flow f_i. E.g. say we are using shortest path routing, and the QoS specifications permit routing the flow along not only the shortest path but also on paths which are having at most two hops more. Then the set $\mathscr{R}_{f_i}$ will contain all the routes which are thus possible. Let us denote the routes in this set as $r_{f_i}^1 \ldots r_{f_i}^{|\mathscr{R}_{f_i}|}$.

$$\text{Constraint } C_1\text{: } V\left(f_i\right) \geq 0 \quad \text{Value of flow } f_i\text{, e.g. } V\left(f_i\right) = b_i \tag{8.5}$$

$$\text{Constraint } C_2\text{: } \rho(r_{f_i}^j) = \begin{cases} 1 & \text{iff } r_{f_i}^j \text{ is the chosen route for } f_i \\ 0 & \text{otherwise} \end{cases} \tag{8.6}$$

$$\text{Constraint } C_3\text{: } A\left(f_i\right) = \begin{cases} 1 & \sum\limits_{r_{f_i}^j \in \mathscr{R}_{f_i}} \rho\left(r_{f_i}^j\right) > 0 \\ 0 & \text{otherwise} \end{cases} \tag{8.7}$$

$$\text{Constraint } C_4\text{: } \sum_j \rho(r_{f_i}^j) \leq 1 \ \forall i \tag{8.8}$$

$$\text{Constraint } C_5\text{: } a_e^m = \begin{cases} 1 & \text{iff edge } e \text{ is active in minislot } m \\ 0 & \text{otherwise} \end{cases} \tag{8.9}$$

$$\text{Constraint } C_6\text{: } d_e = \sum_i \sum_j \rho(r_{f_i}^j) \cdot b_i, \quad \forall\, e \in \mathscr{E} \tag{8.10}$$

$$\text{Constraint } C_7\text{: } \sum_m a_e^m = d_e, \quad \forall\, e \in \mathscr{E},\ \forall \text{ minislots } m \tag{8.11}$$

$$\text{Constraint } C_8\text{: } a_e^m + \sum_{e_x \in I(e)} a_e^m a_{e_x}^m \leq 1, \quad \forall m,\ \forall e,\ e_x \in \mathscr{E} \tag{8.12}$$

$$\text{Constraint } C_9\text{: } 1 \leq m \leq M, \quad M = \text{maximum no. of minislots} \tag{8.13}$$

Constraint C_2 introduces an indicator function $\rho(r_{f_i}^j)$ which takes the value 1 if and only if the flow f_i is routed along route $r_{f_i}^j$. If the flow is not routed along that route or if no route is chosen for the flow (e.g. due to the fact that possibly no feasible route exists).

Constraint C_3 provides the definition for the indicator function $A(f_i)$ which indicates, as already discussed, whether the flow is admitted into the WMN or not. Thus, as specified in Eq. (8.7), $A(f_i)$ will be 1 if one or more routes are selected for routing the flow f_i. However, we want to restrict the routing to single path routing for reasons of avoiding jitter and out of order delivery of packets.

Constraint C_4 together with the definition of $\rho()$ in constraint C_2 ensure that at most a single route from the set of permissible routes is used to route the flow. Furthermore, it is permissible that no route is selected for a flow. Thus it is possible not to admit a flow to the network (e.g. in case its QoS or bandwidth requirements cannot be satisfied).

We use the variable m to denote a minislot. Minislots are numbered from $1 \ldots M$. Where, M is the maximum number of minslots in a frame which are available for scheduling data transmissions. This is also specified by constraint C_9.

Constraints C_5 to C_9 ensure that feasible schedules are found for the flows subject to the interference constraints described in Sec. 8.1.1. In constraint C_5 we introduce variables a_e^m for each minislot m and edge e in the WMN. a_e^m takes the value 1 if and only if the edge e is active in minislot m. If edge e is not active in minislot m then value a_e^m is 0. Constraint C_6 introduces variables d_e for each $e \in \mathscr{E}$, which counts the total demand on edge e. I.e., d_e specifies the total number of minislots which are needed per frame to satisfy the bandwidth requirements of all the flows which are routed using routes which include the given edge e. Constraint C_7 ensures that the edge e is activated for exactly as many minislots as are required for the given link. A link can be understood to be activated in a given minislot if that minislot is allocated for transmissions on the given link. Allocating less minislots than required for a given edge will mean that the bandwidth requirements of flows which are routed using that edge will not be satisfied. On the other hand, allocating more minislots than needed for an edge is possible, however, this will needlessly block these slots for links in the neighbourhood which possibly cannot use these slots for their own transmissions. Hence, allocating more slots to an edge than needed is not permitted, and is also of no benefit.

Constraint C_8 ensures that no two conflicting links are activated in the same minislot. Note that for this we use our definition of $I(e)$ which we introduced in Eq. (8.1). This ensures that the schedules computed are conflict free as per our definition of the interfering links (see discussion in Se. 8.1.1 for details).

Finally, constraint C_9 ensures that we use at most only as many minislots as are available in the system at the maximum. A solution to the above optimization problem provides a route (or none) for each flow as well as a schedule for the flows which are admitted to the WMN.

As can be seen from the constraints $C_1 \ldots C_9$ and the objective function in Eq. (8.4), it is not possible to use simple linear programming approaches to solve the above problem. This is mainly due to the discrete nature of the problem, i.e. we must allocate discrete timeslots (minislots) and cannot allocate arbitrary fractions of the subframe to edges. Furthermore, we cannot split and route a single flow via multiple routes at the same time. This means that discrete routes need to be allocated to flows. This brings us to the domain of Integer Linear Programming (ILP) and Mixed Integer Linear Programming (MILP) problems, which, are generally considered to be NP-hard (see [18] for definition of NP-hard).

We evaluated the proposed optimization problem using the open-source *GNU Linear Programming Toolkit* [29]. GLPK, uses advanced methods such as *branch-and-bound* to efficiently solve such problems. We tested our problem for small network sizes (up to 10 nodes, up to 15 links, up to 5 flows) and very few (up to 10) minislots. For these networks, GLPK was able to provide a solution to the system of equations within a few tens of seconds with reasonable results: i.e. correct routes where found, with a single route per flow, and no collisions occurred. For larger networks (e.g. 16 nodes in a 4x4 grid layout), the solver did not compute any solution even after providing it quite a large amount of time for the computations (e.g. 24 hours). Similar issues arose when we increased either the number of flows or the number of links in the WMN. Although, more sophisticated techniques such as column generation [87] can be used to produce results much faster, we can conclude that such optimization based approaches are highly infeasible for application in realistic WMN deployment scenarios with dynamic changes of traffic. Even with these approaches it is mostly not possible to obtain the needed results in real-time. Note, that for the above problem we haven't added network coding. Adding network coding to the above problem will further increase the complexity. Adding constraints for network coding involves generation of virtual edges in the graph for each network coding opportunity and

additionally specifying the interference and demands on the links. Note, that the virtual edges need to be created dynamically based on the network coding constellation. For example, one virtual link may be added to model the transmission of coded data by the relay nodes. Other virtual links may be added for considering the special interference conditions which need to be met when transmitting uncoded data to the relay which needs to be used for coding at a later stage. Thus, both the set of links as well as the interference constraints for this need to be dynamically determined at runtime. This complicates the problem further, and as the optimization approaches did not prove reasonable for the problem even without network coding, we did not consider this further, and will not be presenting a network coding enabled version of the optimization problem here.

Hence, for CORE we decided to use heuristics to solve the above problem efficiently for real deployment scenarios. We will now look at the dimensions of the problem to be solved to get a better idea about the complexity of the problem at hand.

8.1.3 Complexity of the Solution Space for the Optimal Route Combination Selection Problem

Let us assume that we have the set of flows $\mathscr{F}$ which need to be admitted to the WMN. Then using the optimization goal we discussed previously, the solution to this problem is in essence a vector of routes $(r_1^a,\ r_2^b, ..., r_{|\mathscr{F}|}^t)$. The individual routes $r_{f_i}^k$ for the flows f_i are chosen from the set of permissible routes for the respective flows, i.e. from the set $\mathscr{R}_{f_i}$. In general, finding an optimal solution for (8.4) involves an exhaustive search over the entire solution space. Assume that the MBS knows about a total of $|\mathscr{F}|$ flows and considers K_{max} possible permissible (w.r.t. QoS) paths for each of them. If there exists a valid schedule such that all $|\mathscr{F}|$ flows can be scheduled, the optimum is one of $(K_{max})^{|\mathscr{F}|}$ combinations of routes which are possible. If the $|\mathscr{F}|$ flows are not schedulable at the same time, one has to drop at least one flow. Assuming that $(|\mathscr{F}| - i)$ flows are schedulable together (i.e. we drop i flows), the optimum route combination is one of the $\binom{\mathscr{F}}{\mathscr{F}-i} \cdot (K_{max})^{(|\mathscr{F}|-i)}$ feasible route combinations.

Usually we do not know how many flows and which ones can be jointly routed such that a valid schedule can be found. Therefore, one has to additionally determine the maximum set of schedulable flows. This can be done by starting with all $|\mathscr{F}|$ flows and removing flows one-by-one until a valid schedule is found for the remaining flows, or by starting with a single flow and adding further flows until the maximum set of schedulable flows is reached or via some kind of binary search. In the worst case, one checks each possible combination of routes for every subset of flows. The upper bound on the number of possible combinations thereby can be calculated as shown in Eq. (8.14).

$$Comb_{max} = \sum_{i=0}^{|\mathscr{F}|-1} {K_{max}}^{|\mathscr{F}|-i} \begin{pmatrix} |\mathscr{F}| \\ |\mathscr{F}| - i \end{pmatrix} = \left(1 + K_{max}\right)^{\mathscr{F}} - 1. \tag{8.14}$$

For clarification we give two numerical examples:

1. If there are $|\mathscr{F}| = 2$ flows, each having $K_{max} = 5$ possible routes, there are $c = 5^2 + 2 \cdot 5^1 = 35$ combinations. Where, there are 25 combinations possible in which both flows are scheduled, and 10 possibilities where only one of the two flows is routed. Which is the same number which we will obtain using Eq. (8.14).
2. If the number of flows is increased to $|\mathscr{F}| = 10$, with K_{max} remaining 5, there are $c = 6^{10} - 1 \geq 6 \cdot 10^7$ combinations.

Note, that in order to determine if a given combination of routes for the given flows is feasible or not we need to check if there is a possible feasible schedule for the demands of the flows. Thus, in order to just determine if a route combination is schedulable or not for the flows in question a full transmission schedule for the WMN needs to be computed and tested. This subproblem alone is an NP-Hard problem ([17, 62]).

It is possible to use advanced and specialized algorithms such as nonlinear column generation [44] to subproblems for problem (8.4). However, the computation time required is still unacceptable for problem sizes encountered in real networks. Hence, we are interested in finding algorithms that use heuristics to find a near-optimal solution to the problem with reasonable computational effort. Hence CORE uses heuristics to approach its optimization goal.

CORE uses both centralized as well as distributed components to achieve its goal. In Sec. 8.2 we present an architectural overview of its components and an overview of the interaction between the individual components and their functions. In Sec. 8.3, Sec. 8.4, and Sec. 8.5 we present the heuristics used by CORE corresponding to the respective phases of operation as discussed in Sec. 7.2. These heuristics together aim to find a solution for the problem (8.4). The heuristics additionally support the use of network coding.

8.2 CORE Architecture

We looked at the optimization problem CORE seeks to address in some detail in Sec. 8.1. We saw that an optimization based approach though feasible in theory, falls short of the requirements for deployment in practice. For one, the computation of the solution takes too long to be of any use. Secondly, this would require complete centralized control of the scheduling and routing in the wireless mesh network. Thereby making the WMN more susceptible to failures of the central node carrying out the optimization. CORE uses heuristics based to approach the optimization goal. Although, CORE uses centralized computation of the heuristics, care is taken in the design of the CORE architecture to ensure that the WMN is not brought down by a failure of the central server running CORE's heuristics.

In this section we look at the architecture of CORE in detail. We provide an overview of the individual components involved in the CORE framework, and their position in the protocol stack of a node in the WMN*.

Fig. 8.2 shows a wireless mesh network with a typical scenario which we assume for the rest of the discussion. As shown in Fig. 8.2 the WMN is supposed to be composed of a set of nodes (subscriber stations). A given node in the WMN (called the mesh base station) provides the nodes in the WMN access to external networks and the Internet via either a wired or wireless link, which is not considered to be a part of the WMN.

As discussed earlier, CORE runs heuristics centrally. Let us call the node which runs the CORE heuristics and is responsible for the maintenance of CORE's logical functioning throughout the WMN as the central server. Thus, the central server or CORE server is a node at which CORE's heuristics are run. For practical reasons we assume and recommend that the CORE server be located at the mesh base station. However, the CORE server can be any node in the WMN. As shown in Fig. 8.2, we assume that the CORE server is the same as the mesh base station. The CORE framework is designed such that, although, the heuristics are computed centrally, the deployment of the computed solution relies on distributed components. Thus, the subscriber stations (nodes) in the WMN too play a vital role in the CORE framework. CORE makes use of the distributed components which are assumed to be present in a reservation based WMN supporting network coding. Thus we assume that each node supports means for scheduling,

* Assuming that CORE is supported by the nodes in the WMN.

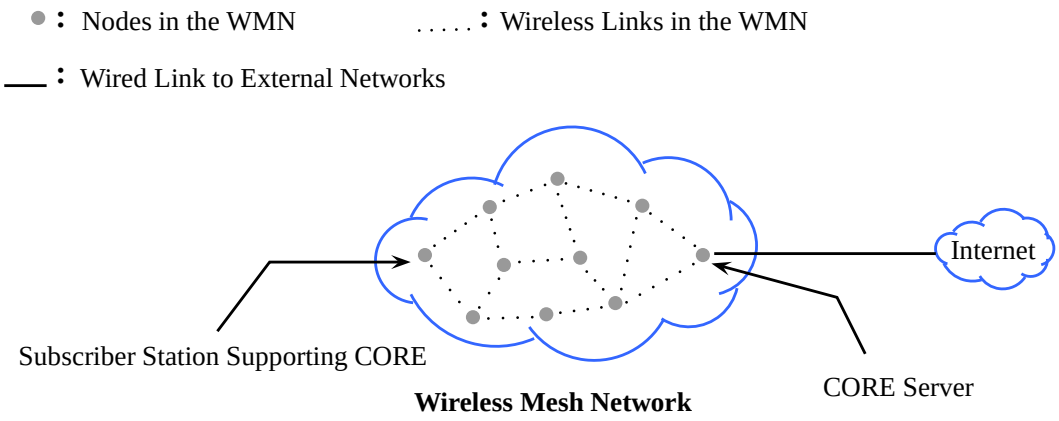


Figure 8.2: Wireless Mesh Network and CORE's Components

bandwidth reservation, and also supports routing as well mechanisms for network coding. To study the concept in concrete settings, as previously, we assume that the nodes in the WMN use the IEEE 802.16 MeSH mode. Further, we will assume that each node supports our stream-oriented network coding (SONC) mechanisms proposed in Part II of this thesis.

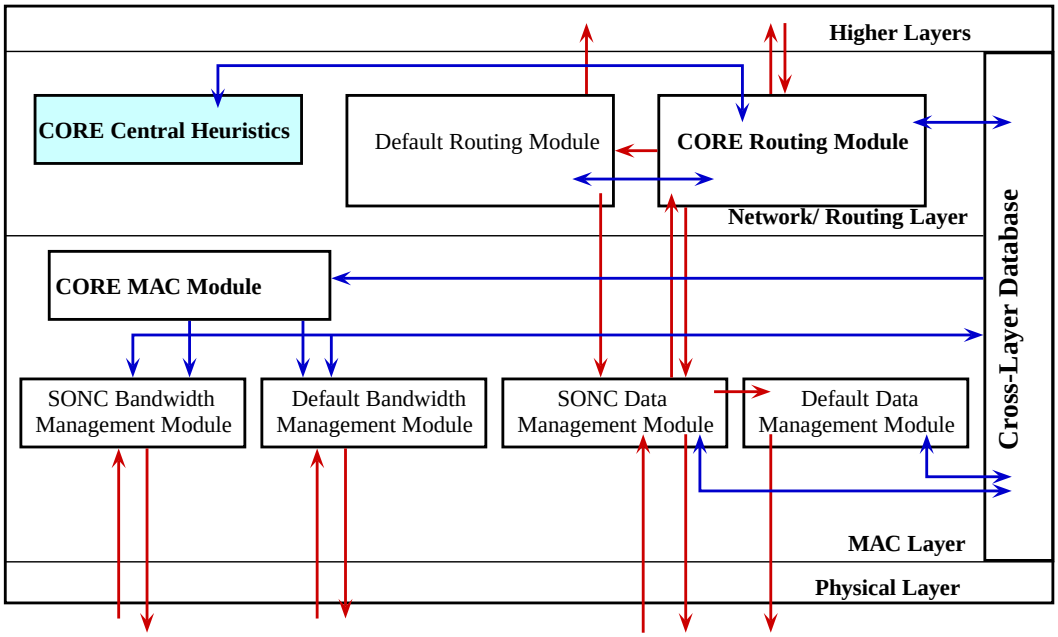
Fig. 8.3 shows the protocol stack architecture of a node supporting CORE. In Fig. 8.3 we show only the main building blocks, putting them in their logical position in the protocol stack keeping the well known Transmission Control Protocol/Internet Protocol (TCP/IP), or the OSI Protocol stacks as a reference (see [56, 112] for details about the TCP/IP or OSI reference protocol stacks). The protocol stack layers shown in Fig. 8.3 correspond to the physical, datalink and network layers of the two reference protocol stacks.

As the title of the thesis suggests, CORE uses a cross-layer approach to enable better coordination among the otherwise completely independent protocol layers. This is done in order to ensure that the layers do not initiate contra productive actions by the nodes. As shown in the figure, CORE's components can be logically placed in the medium access control (MAC) layer and the network or routing layer. For the purpose of the discussion we will focus on these two layers. The physical layer is assumed to be as defined in Sec. 1.1.

In Fig. 8.3, the arrows depict either internal flow of control information between the different modules, or the flow of data packets which are received from the wireless medium or are eventually scheduled transmission on the wireless medium. The two types of flows are differentiated by the different colours used. Blue arrows are used to depict internal flow of control information, and red arrows are used to depict flow of data or control packets. The arrow-heads show the direction of the information flow. Bidirectional arrows indicate that information or packets flow in both directions between the corresponding modules. Modules with a shaded background indicate modules that are active only at the CORE server. These may, however, be present at all the nodes in the WMN. We will now first look at the individual modules, looking at the functionality these provide (Sec. 8.2.1). This is followed by a discussion (in Sec. 8.2.2) spanning the lifetime of sample data flows in the WMN, as observed at a given node, highlighting the interaction of the individual modules. The goal of the latter description is to highlight the key events which trigger actions at the individual modules involved in the CORE framework, and how these interact with each other to give the complete picture of the running system.

: Flow of packets (control/data)

: Flow of internal control Information



Protocol Stack of the Nodes in the WMN showing Vital CORE Components*

(*Note: components present only at the CORE Server are shown shaded)

Figure 8.3: CORE's Main Components and Their Logical Position in a Node's Protocol Stack

8.2.1 CORE's Modules

We now list the important components involved in the CORE framework which are depicted in Fig. 8.3 highlighting the role played by each component in the CORE framework.

- CORE Central Heuristics Module: This module is shown shaded in Fig. 8.3. It is only actively needed at the CORE server. It may be present at all the nodes in the WMN. However, the module should only be active at the CORE server. The task of this module is to receive and process the CORE messages it receives from all the nodes in the WMN (See Appendix B for details of the CORE messages). For example it receives and processes messages indicating changes in the traffic flows originating at individual nodes in the WMN. On receiving these updates, the heuristics module uses the heuristics presented in Sec. 8.4 to compute the routes to be taken by individual flows in the WMN. The heuristics module then generates appropriate route update messages which are then sent to the corresponding nodes in the WMN. In case this module fails at the central server, or the central server itself fails, then the CORE messages being sent by the nodes will not be processed. This in turn implies that the CORE framework will not initiate any route changes in the WMN irrespective of the current state of the flows in the network. In this case the rest of the WMN can continue normal operation using the default routes to be taken for the individual flows as per the current state of the routing tables at the individual nodes.
- Default Routing Module: We assume that this module represents the standard routing algorithm being used at all the nodes in the WMN. Typically for a WMN, either link state or distance vector routing algorithms, which proactively maintain the routing information for all destinations in the WMN are meaningful. As its name suggests, CORE only provides extended functionality to the default routing module, adapting the default routes with routes as directed by the central server. If the CORE routing module at a node does not have any routing information for a flow then the default route in the routing table maintained by the default routing module is used. This route is usually the shortest hop path to the destination node in the WMN.
- CORE Routing Module: This module is responsible for routing packets belonging to individual flows on routes as directed by the CORE server. This module is present at all the nodes in the WMN. The CORE routing module when present at a node in the WMN extends the functionality of the default routing module present at the node. Initially, the CORE routing module uses the routing information available from the default routing module to decide the next hops for all the flows which pass through the node. However, once the CORE server has valid routing information, for a set of flows in the network, provided by the CORE server, then this routing information is used to route packets belonging to these flows.

  For example, say the CORE routing module at a node n_i obtains routing information for a flow $f(s_j, d_j)$ from the CORE server. Assume that the routing information from the server instructs the node to use n_j as the next hop for all packets arriving at that node belonging to the flow $f(s_j, d_j)$. Suppose that the default next hop (computed by the default routing module) for packets going to destination d_j at node n_i is neighbour n_d. Then until the node receives the above routing information from the CORE server the default next hop will be used for all packets to be routed to destination node d_j. However, once the node has routing information from the CORE server pertaining to flow $f(s_j, d_j)$, then, packets with destination d_j arriving from the source node s_j will be routed to the next hop n_j. For the packets arriving at node n_i with destination d_j which do not belong to the flow $f(s_j, d_j)$ the default next hop (i.e. node n_d) will be used.

Besides the above routing functionality, the CORE routing module is also responsible for periodically obtaining information about the flow statistics from the cross-layer database. This database contains information about the flow such as the average traffic arrival rate, the average bandwidth required for sending the packets for a flow etc.. Additionally, the database also stores information such as the current queue occupancies at a node for each outgoing queue. For more details about the kinds of statistics which can be maintained see Ref. [84]. These statistics are best computed at the MAC layer, also the reservation based MAC layer can be assumed to measure and use these statistics anyway for estimating the bandwidth it should reserve. As stated, the CORE routing module observes these statistics and notifies the CORE server in case it notices significant changes (see Sec. 8.3 for more details). Thus, the CORE routing module performs the role of providing extensions to the existing routing in the WMN.

It also processes and handles CORE related control messages arriving at a node. Based on the information obtained from the CORE messages as well as its own observations, it also initiates updates to elements in the cross-layer database. It is also responsible for generating CORE control messages periodically to keep the CORE server informed about the state of the flows originating at the node.

- CORE MAC Module: The CORE MAC module is present at all the nodes in the WMN. It is responsible for handling the CORE Framework specific extensions at the MAC layer. The CORE MAC module, in an event based manner as well as periodically observes the information stored in the cross-layer database by the CORE routing module. In accordance with the information obtained from the CORE routing module it initiates and triggers control messages to be sent by the SONC bandwidth management module, as well as the default bandwidth management module at a node. The CORE MAC module does so by providing appropriate control information to the two bandwidth management modules.
  For example, say the CORE routing module has obtained route updates from the CORE server. It will then activate the new routes at the node at the selected frame. However, switching the route alone for an existing flow is not sufficient. If this alone is done, then packets for that flow would have to wait in the output queue for the new next hop till sufficient bandwidth is reserved on that link. The delay will be especially exorbitant if no bandwidth at all has been previously reserved for transmissions on the link where the flow has now to be rerouted. To avoid such QoS degradation the CORE framework foresees that the nodes are notified of the routing changes in advance, and given that they know the frame in which the routing changes will be affected, the nodes should also reserve bandwidth in advance in correspondence to the updated bandwidth requirements due to the routing updates. Thus, it means that more bandwidth needs to be reserved for some links (where an increase in bandwidth requirements is expected) and bandwidth reserved for transmission on some links needs to be freed or reduced (for links where the traffic demand will decrease as a consequence of the route updates). More details about this aspect can be found in Sec. 8.5.
- SONC Bandwidth Management Module: This module corresponds to the bandwidth management module for network coding in reservation based WMNs. Details of its working were presented in Part II of this thesis. Here, we will only briefly outline the role it plays in the CORE framework. As shown in Fig. 8.3, the SONC bandwidth management module receives additional control information from the CORE MAC module regarding the bandwidth reservation changes. It may then react on this information to adapt its bandwidth reservations.
- Default Bandwidth Management Module: This module is the default bandwidth management module for a node. For the current thesis we assume that it works as in Ref. [84]. However, other implementations for this module are also possible and can be incorporated

into the CORE framework. This module is responsible for reservation of bandwidth for the normal unicast links in the WMN. In addition to the information about the bandwidth requirements for each neighbouring link, which it obtains from the statistics maintained in the cross-layer database, the module also receives control information from the CORE MAC module. On receiving this information it may adapt the bandwidth reservations for the individual outgoing links at the node. Details about this aspect can be found in Sec. 8.5.

- SONC Data Management Module: This module is the module which handles packets for flows which are involved in network coding sessions currently. The functioning of this module was explained in detail in Part II of the thesis. By default, when nodes support CORE as well as network coding, all data packets will be initially handled by this module in both directions (i.e. from the higher layer to the lower layers as well as vice versa).
- Default Data Management Module: This module is the default data management module assumed to be present at nodes even in the absence of network coding. The functionality of this module is similar to that provided by the SONC data management module, i.e. queueing of packets for transmission, fragmentation, defragmentation, scheduling of the packet transmissions etc. It however, does not provide any functionality needed for supporting network coding.

These are the main modules of interest when discussing the CORE framework. As seen from Fig. 8.3 we build up on our SONC approach presented in Part II of this thesis. The additional functionalities for global (i.e. WMN wide) optimization of routes and the corresponding bandwidth reservation changes are provided by the CORE Central Heuristics Module, the CORE Routing Module, and the CORE MAC Module. Thus these three modules will also be the focus of the following sections. In this section we had a look at the main modules, and provided an overview of the roles played by each of them in the CORE framework. In the next section, we provide more insights into the functioning of the CORE framework as a whole, and the interaction of the individual modules by following data packets belonging to selected sample flows in a sample WMN topology. We thereby look at how the data packets flow between the individual modules, and the control information, and control packets generated when significant events occur. This will help the reader to better understand the interaction of the modules and the CORE framework as a whole.

8.2.2 Data Flow and Interaction Between Modules in the CORE Framework

Suppose we have the WMN topology shown in Fig. 8.4 with five nodes labeled 1 ... 5. Assume that, for the sake of keeping the discussion simple, that we have only two flows in the WMN, namely, $f(5,1)$, $f(1,3)$. As shown in the graph in Fig. 8.4, flow $f(5,1)$ enters the WMN at time t_2, and stops at time t_5. Flow $f(1,3)$ enters the WMN at time t_1 and stops at time t_2. Thus, for the time t_2 to time t_5 both the flows are present in the WMN simultaneously. As per our notation, $f(5,1)$ has source node 5 and destination node 1 and $f(1,3)$ has source node 1 and destination node 3.

Let us assume for the sake of simplifying the discussion, that both the flows have constant and exactly the same bit rates. Further, let us assume that both flows are detected as long-term flows with a stable data rate in ϵ time after their arrival in the WMN. Thus, for simplicity we assume that the CORE routing module detects that each flow is having a steady data rate and gets updated statistics about the individual data rates from the cross-layer database in negligible time (ϵ). Thus, at time $t_1+\epsilon$ the CORE Routing Module at the node 1 will detect that it has a stable flow $f(1,3)$ and generate control packets to notify the CORE server of the new demand arriving at the node in the WMN. These control packets are then handed over to

the lower layer for transmission with the destination for these packets being the CORE server. At intermediate nodes on the path to the server, these demand update control messages will be handled by the CORE Routing Modules at the individual nodes and then routed to the next hop on the path to the server. Once the CORE server receives these control messages, it will update its local statistics, as well as inform the CORE Central Heuristics Module of the change (addition/change in bandwidth/dropping) of flows in the WMN. Thus, in the CORE framework, the CORE Routing Module at the source node is responsible for keeping the CORE server up to date about the bandwidth requirements and flows which originate at that node.

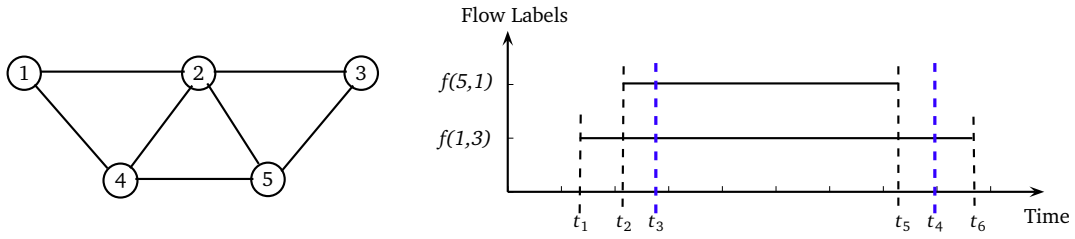


Figure 8.4: Sample WMN Topology, with Flows Between Nodes 5 and 1 and Nodes 1 and 3 ($(f(5,1)$ and $(f(1,3))$) with Corresponding Lifetimes of the Individual Flows

For the sake of discussion let us assume that the CORE server is present at node 2 and also provides the nodes in the WMN access to external nodes and the Internet. Thus, in our simplified example the CORE server (node 2) will be notified of the existence of flow $f(1,3)$ and its bandwidth requirements at time $t_1+\epsilon$ the server will get updated information about flow $f(5,1)$ at time $t_2+\epsilon$. Thus, at time instant $t_2+\epsilon$, the CORE Central Heuristics Module[†] at the CORE server will have information about both the flows which exist in WMN.

Assume that per-default we are using shortest hop routing in the WMN, and that the path $1 \rightarrow 2 \rightarrow 3$ is being taken by packets belonging to flow $f(1,3)$ and the path $5 \rightarrow 4 \rightarrow 1$ is being taken by the flow $f(5,1)$. Now, at time instant $t_2+\epsilon$, the CORE Central Heuristics Module will process the flow information it has obtained to detect that it can save bandwidth via network coding and reduce interference in the WMN as a whole by routing flow $f(1,3)$ along the slightly longer path $1 \rightarrow 4 \rightarrow 5 \rightarrow 3$. This will enable node 4 to deploy network coding and it will also reduce the data flow through the highly connected node 2 leading to less blocked links in the WMN for the same data transmission. Thus, at time $t_2+\epsilon$, based on the solution provided by its heuristics, the CORE Central Heuristics module will determine that flow $f(1,3)$ should be routed along path $1 \rightarrow 4 \rightarrow 5 \rightarrow 3$, and flow $f(5,1)$ should be routed along path $5 \rightarrow 4 \rightarrow 1$.

The CORE Central Heuristics Module then generates appropriate control messages to inform the nodes in the WMN of the routing changes to be affected, and the corresponding bandwidth reservation changes. These control messages will then be handed over to the CORE Routing Module at the CORE server to be routed to each of the affected nodes. In our example, these are the nodes 1, 2, (along the old route for flow $f(1,3)$) and the nodes 1, 4, and 5 which are on the new route for the flow $f(1,3)$.

In our example only one flow is affected by routing changes hence only nodes which were involved in the data transmissions for this flow in the past, and such which shall be involved in the data transmissions for the flows with route changes in the future. Note, that in realistic WMNs it is not possible to deploy the route changes as well as bandwidth reservation changes involved instantaneously. This is due to the fact that the control packets sent by the CORE cen-

[†] To maintain readability of the discussion, we will assume that the CORE Central Heuristics Module is only present at the CORE server, and hence when referring to it we are always referring to the module at the central server.

tral server will require a certain number of frames till they arrive at all the nodes addressed by the control messages. Further, the nodes on receiving these messages need to process these messages and start the bandwidth reservation negotiation to affect the bandwidth reservation changes needed. The bandwidth reservation process itself takes some time, and even on successful bandwidth reservation the node may have to wait a certain number of frames before it can use the reserved bandwidth, as the bandwidth can be granted up to a certain number of frames in the future. In realistic WMNs these individual delays are variable and not negligible. Let us assume that after the CORE Central Heuristics Module issues the route and bandwidth update control messages a time duration δ_{Ctrl} is needed till all the affected nodes receive and process the control messages. Further, let δ_{BwRes} denote the additional time needed for all the affected nodes to complete their bandwidth reservation changes, and the involved negotiation, and the time till they are able to actually get bandwidth reserved as indicated by the control messages from the CORE Central Heuristics Module.

Thus, in our simplified example at time instant $t_2 + \epsilon + \delta_{Ctrl} + \delta_{BwRes}$ the process discussed above will be complete, and all the nodes will be able to have sufficient bandwidth reserved on the new planned routes for the flows in the WMN. The CORE framework at runtime maintains running estimates for these individual delays which can be expected. If the operator provides preset values for these variables based on observations and experience, these can be used instead too. In practice, especially in IEEE 802.16 based WMNs, when the CORE server is placed at the MBS then there will be bandwidth reserved for transmission on the tree rooted at the MBS to the individual SSs in the WMN. This is because the MBS provides the WMN with access to external networks and the Internet, and thus, as usually all subscribers will have Internet traffic originating at the nodes or for which they are the sink, sufficient bandwidth will usually be reserved for this traffic on the links on the tree to the MBS. In the IEEE 802.16 setting, this bandwidth will usually be reserved via centralized scheduling (see [42, 83] for details). However, even if centralized scheduling is not being used in the WMN, the necessary bandwidth can be reserved using distributed scheduling, albeit at a slightly larger control overhead. In our implementation of the node, we prioritized the control messages sent by CORE over the normal data traffic to be transmitted, hence, when even a small amount of bandwidth is available for transmission on the links of the scheduling tree rooted at the MBS, the CORE control messages generated by the CORE Central Heuristics module at the MBS reach the SSs with a negligible delay. The same is the case for the delays for the CORE control messages generated by the CORE Routing modules at the individual SSs.

In our example we saw that when the CORE server issued the route update control messages at time $t_2 + \epsilon$, by time $t_2 + \epsilon + \delta_{Ctrl} + \delta_{BwRes}$, the nodes in the WMN will have received the control messages, and will have had sufficient time to complete the corresponding changes to their bandwidth reservations. It makes sense to activate the new routes and switch from the old routes to the new routes only after this time, else, it may lead to huge delays due to the missing bandwidth reservations on the new routes. Let us denote the time $t_2 + \epsilon + \delta_{Ctrl} + \delta_{BwRes}$ as time t_3. The CORE framework in practice chooses to activate the new routes at all the nodes at exactly the frame for time instance $t_2 + \epsilon + \delta_{Ctrl} + \delta_{BwRes} + \Delta$, where the additional time interval Δ provides a safety margin for unexpected delays or packet loss. The operator can, based on prior experience, choose appropriate values for these parameters. In our example we will assume that Δ is negligible. Thus, the CORE Central Heuristics module can choose the time instant (frame containing this time instance) t_3 as the time instance for activating the new routes at the nodes in the WMN. Thus, in our example, at the frame corresponding to t_3 the CORE Routing modules at the nodes in the WMN will all simultaneously update their local routing tables such that the packets for flow $f(1,3)$ are routed along path $1 \rightarrow 4 \rightarrow 5 \rightarrow 3$, and the packets for flow $f(5,1)$ are routed along path $5 \rightarrow 4 \rightarrow 1$. The bandwidth necessary for the packets for these flows will have already been reserved on all the links along the new

paths, and the bandwidth on the old routes which is no longer needed will have already been freed. The CORE MAC module is responsible for effecting these bandwidth reservations and changes to the previous bandwidth reservations based on the information it obtains from the cross-layer database. The cross-layer database in turn will have this information provided by the CORE Routing modules at the individual subscriber stations. The CORE Routing module on its part obtains the information about the routing changes and the corresponding bandwidth reservation changes from processing the control messages sent by the CORE Central Heuristics module. Thus, starting at time t_3 the new routes will be used for the flows in the WMN.

After time t_3, the SONC module at node 4 will realize that it has a set of flows in a constellation which is conducive for network coding. In practice, the SONC module will not initiate the setup of bandwidth for network coding for an additional time till it can decide whether the flows are stable enough (for details see Part II of this thesis). Here, we consider this additional time duration to be negligible. Thus, in our example from Fig. 8.4, we see that at time t_3 the SONC module at node 4 will decide that network coding is feasible for the two flows flowing across the node, and will initiate the bandwidth reservations for network coding, and will start maintaining data and packets for coding and decoding of the coded packets one SONC is active for these flows. Using similar arguments, at t_5 the SONC module at node 4 will realize that network coding is not possible for the given flows any more. This is as the flow $f(5,1)$ has stopped. Also, at time t_5, the SONC module at node 4 will stop the network coding session, and free the bandwidth reserved for the network coding session. In practice, after the application at the source stops sending any data it will take some number of frames till the nodes on the path realize that the flow has stopped, or has drastically reduced bandwidth requirements. This delay also depends on the number of packets still in transit in the network belonging to the flow, and the bandwidth which could be reserved for the flow on the individual links along the path to the destination.

The flow change information, i.e. the information that flow $f(5,1)$ has exited the network will be sent at time t_5 to the CORE server by node 5. Now, as discussed previously, the CORE Central Heuristics module on the server will note that the routing change it has previously initiated (at time t_3) is no longer meaningful, as it does not bring any gain via network coding to the WMN. Hence, the shortest routes with lesser interference will be used. Say, in our simple example, that the CORE Central Heuristics module makes the decision to change the routes at time $t_5 + \epsilon$. Then as in the previous discussion, it will take some additional time for the control messages about the routing changes to reach the nodes in the WMN, and some further time will be needed till the bandwidth reservations in the WMN are adapted in accordance to the new routes. So estimating this delay, in our example, the CORE Central Heuristics module instructs the nodes in the WMN to effect the routing changes at frame corresponding to time instance t_4. Thus, after time t_4 the only remaining flow in our example, $f(1,3)$ will be routed along path $1 \rightarrow 2 \rightarrow 3$ with no network coding possible. Also at time t_6 the CORE Central Heuristics module will be notified by node 1 about the end of flow $f(1,3)$. The CORE Central Heuristics module will then generate appropriate control messages to cancel any fixed bandwidth reservations it had instructed the nodes to keep alive for flow $f(1,3)$.

To summarize the above discussion we see the following major events as far as the CORE framework is concerned during the lifetime of the two flows in the WMN. Slightly after time t_1 the CORE Central Heuristics module obtains information about the flow $f(1,3)$. At a time just after time t_2 the server receives information about the flow $f(5,1)$. After obtaining this information the CORE Central Heuristics module computes new routes for the flows which will lead to bandwidth savings via the deployment of network coding, and also will reduce the total interference, i.e. number of blocked links, in the WMN. By the time t_3 the nodes in the WMN will have made preparations for the routing changes, as triggered by the control

messages received by the nodes from the CORE Central Heuristics module. Thus the nodes in the WMN will be able to activate the new routes starting at after time t_3. Following this, in a short interval of time, the SONC module at node 4 will detect a viable network coding constellation allowing it to code together packets belonging to the two flows existing in the WMN. At time t_5 the CORE Central Server is notified by node 5 that flow $f(5,1)$ no longer exists in the WMN. After getting this information the CORE Central Heuristics module will compute new routes, and corresponding bandwidth reservation changes which then need to be effected by the nodes in the WMN. Allowing for sufficient time for all nodes to be notified, and for the bandwidth negotiations to complete, at time t_4 the new routes and the corresponding bandwidth reservations will be activated by the nodes in the WMN. Finally, at time t_6 the CORE Server will be notified by node 1 about the exit of flow $f(1,3)$ from the WMN. This will then lead to appropriate control messages for further routing updates, and bandwidth reservation changes to be sent by the CORE Central Heuristics Server, if needed.

Thus, in this section we saw in quite some detail how the CORE framework works over the lifetime of flows in the WMN. We thus see that the CORE framework encompasses vital components at the MAC and the networking layers, and is thus essentially a cross-layer approach to enhancing the gains which can be obtained by SONC in reservation based WMNs. As we have seen, the centralized control of the WMN is only loose, and the CORE server only instructs the nodes as to the actions these have to take (especially with respect to routing, and bandwidth reservations). However, the implementation of these instructions is left up to the distributed components. Network coding for one is carried out completely in a decentralized manner, with the SONC mechanisms presented in Part II of this thesis being applied. This makes the CORE framework more robust to failures of the central server than a network where the entire schedules, routes as well as the network coding is steered by a central entity. This however does not imply that complete central control of the transmission schedules in the WMN is not possible, and the mechanisms we have presented can be easily extended, by putting in other modules in the CORE framework which wait for complete instructions as to the transmission schedule and routing and network coding from the central server.

In the next few sections (Sec. 8.3, Sec. 8.4, and Sec. 8.5) we will look at the details of the heuristics used by the CORE framework in the different logical phases of operation explained in Sec. 7.2.

8.3 Phase I: Heuristics for Network State Observation and Reporting to Central Server

As discussed shortly in Sec. 7.2, CORE can be considered to be operating in three logical phases, which are by no means strictly sequential in time in real WMNs. In this section, we will first present the goals of the heuristics deployed in Phase I of CORE, followed by a discussion of the implementation we have used for these heuristics. This, is however, by no means the only implementation possible, and the CORE framework permits other heuristics with similar functionality to be used instead.

In Phase I CORE logically proposes to observe the state of the network, especially the flows in the network. By network state we primarily refer to the individual flows in the network as well as the bandwidth requirements for each flow. This information is used later by the CORE Central Heuristics module to compute routes for the flows currently present in the WMN. Without having all the relevant information about the individual flows in the WMN it is not possible for CORE to meaningfully compute feasible routes for the individual flows. In our solution, we propose that the source node for a flow is responsible for periodically reporting information for the flow to the central server. Thus, when a flow exists or arises at a node an update is sent to the central server using the CORE-DEMUPDT message as shown in Sec. B.5.

The use of the message and its fields is discussed in detail in App. B. Here, we will give an overview of the process and discuss the justifications for our design choices made.

8.3.1 Design Considerations and Requirements for the Phase I Heuristics

CORE aims to keep the running overhead of the system to a minimum possible level, as well as reduce the computational complexity. Hence, as also discussed earlier, though the heuristics are run centrally with knowledge about the flows collected at the central point (using defined control messages, see Sec. B.5), the deployment of the solution is left up to the components of the CORE framework operating in a distributed manner throughout the WMN. Referring back to the discussion in Sec. 8.2, Sec. 7.2 and Sec. 8.2.2, we saw that CORE only computes the optimal routes to be taken for the individual flows centrally. It then notifies the resulting route changes (if needed) to the nodes in the WMN, and also provides the nodes with the exact time when the new nodes are to be activated. In addition, the CORE server intimates the nodes about impending bandwidth reservation changes needed corresponding to the route changes.

Looking at the mechanism used it should be clear at first sight, that if the CORE server uses such a process to react to instantaneous bandwidth changes in the individual flows it will be a huge overhead and involve a very large number of route changes, as well as also involve and exorbitantly high overhead due to the large number of bandwidth reservation changes as well as the resulting delays for packets.

Thus, CORE should take care to use the central optimization mechanisms only for flows which are relatively long-lived and only issue bandwidth reservation notifications which cater to the long-term bandwidth demand of the flows. Thus, we see that it is vital to restrict the changes only to long-term flows, as well as consider bandwidth reservations to be directed centrally for the long-term demands.

8.3.2 Identification of Flows and Bandwidth Requirement Measurement

Each node in the WMN maintains records for the traffic originating at the node. In our solution, we propose that each node in the WMN maintain statistics for the bytes arriving for transmission to given destinations at the node from the higher layers. The heuristics for Phase I are run at the CORE Routing Module (see Fig. 8.3). The module identifies based on the source of packets arriving at the module if they are originating at the node itself. If yes, then for flow originating at the node, for each separate destination of the flows, the node will maintain statistics of the arrival rates of the data. This is done by maintaining for example a history of the cumulative bytes originating for transmission to a given destination in each frame. This, in essence gives a time series (see [13] for definition of a time series). Where each sample corresponds to a particular frame and gives the total number of bytes originating at the node for transmission to a given destination. Thus, at each node in the WMN a history of multiple such time series is maintained in the cross-layer database[‡]. Maintaining the statistics in the cross-layer database permits also access to these statistics from the MAC layer which is needed for some bandwidth requirement estimations at the MAC layer in order to reserve the necessary bandwidth.

Let $B_d^s(t)$ represent the total bits of data originating at node s with destination d in frame t. In order to avoid the influence of short term fluctuations in the data arrival rate of the flow, each

[‡] Note that such time series statistics are also needed to be maintained by the MAC layer reservation schemes which we have used for our tests in this thesis, which are however beyond the scope of this thesis. Details of the same can be found in [84]

node computes a moving average of the bits per frame arrival rate. This is computed as shown in Eq. (8.15).

$$Lt_d^s(t) = \frac{\sum_{i=0}^{M-1} B_d^s(t-i)}{M} \tag{8.15}$$

Where, $Lt_d^s(t)$ represents an estimate of the long term data arrival rate per frame (i.e. bandwidth demand) for traffic originating at node s with destination d at frame t. This is the bandwidth demand that is reported by the CORE Routing Module at the nodes to the server using the CORE-DEMUPDT (core demand update) messages. The parameter M can be set to an appropriate value by the network operator. Additionally, bandwidth updates are sent periodically as well as when significant changes to the long-term bandwidth are detected. The period between updates is a parameter which can be set to a suitable value by the network operator based on the targeted preferences. A shorter update period implies more overhead, however, it also permits bandwidth reservations to be directed centrally (long-term bandwidth reservations are controlled centrally by the CORE Server, see Sec. 8.5 for details). A threshold defines whether the change in long-term bandwidth demand is considered to be significant or not. Let us denote this the difference between the currently measured long-term bandwidth requirement and the previously reported bandwidth demand as $\Delta(Lt_d^s(t))$, which is given by Eq. (8.16).

$$\Delta(Lt_d^s(t)) = Lt_d^s(t) - Lt_d^s(t_x) \tag{8.16}$$

Here, $Lt_d^s(t_x)$ denotes the previous long-term bandwidth demand which was reported by the node s to the CORE Server. Now if the absolute value of $\Delta(Lt_d^s(t))$ is greater than the selected threshold value then the node will send a demand update message to the CORE Server informing it of the significant change in the bandwidth requirements of the flow even though the minimum interval between the sending of demand updates has not been exceeded. This permits the CORE Server to react faster to significant changes in the long-term bandwidth demands of flows in the WMN. Thus, the CORE Framework is quite flexible and can be parameterized as desired by the network operator. Additionally, the demand update message presented in the App. B is just a prototype, and in addition to the bandwidth requirements of the flow, one can also consider transmitting further QoS specifications for the flow based on information provided by the application layer for that flow.

In the next section, we will look at the heuristics run at the CORE Server using the information provided by the heuristics presented in this section.

8.4 Phase II: Details of CORE's Central Optimization Heuristics

In Sec. 8.1.2 and Sec. 8.1.3 we saw that an optimization based approach for achieving the aim of CORE (as discussed in Sec. 7.1) is intractable in real dynamic operating scenarios. Here, it is important that we are able to reach CORE's multiple goals in a short duration (i.e. near real-time solution computation is needed). Hence, the CORE Server uses heuristics for the subproblems which need to be tackled in order to meet the optimization goal. To make the optimization problem tractable, and to enable the design of heuristics for finding a solution in real-time, we split the optimization problem into several subproblems. These are: Route Selection and Filtering Heuristic (*RSFH*), Optimal Route Combination Heuristic (*OptRC*) and Maximal Scheduling

Heuristic (*MaxSch*). In the following we will look at each of these heuristics in detail. During the discussion we will also highlight how each of the above heuristics interact with the others, as well as how each of these heuristics falls into place to fill in the gaps towards solving jointly the optimization goal of CORE.

8.4.1 Route Selection and Filtering Heuristic (*RSFH*)

RSFH is used to limit the routes which would be considered as candidates to route a flow entering the WMN to a given maximum. The aim hereby is to keep an upper bound on the computation which would be necessary for the Optimal Route Combination Heuristic (*OptRC*) specified in Sec. 8.4.2, and for the Maximal Scheduling Heuristic (*MaxSch*) specified in Sec. 8.4.3. Normally when a new flow attempts to access the WMN, the flow needs to be admitted to the network by the entity responsible for management of bandwidth in the network. Usually in IEEE 802.16 based network this task is undertaken by the base station. We propose running the centralized component of CORE (i.e. the CORE Server) at the mesh base station. We propose for convenience that the CORE server will be running on the MBS. When the WMN is built with nodes supporting CORE, the node where the flow enters the WMN (i.e. the source of the flow in the WMN) collects the information about the QoS requirements of the flow from the application, and sends this information to the CORE server, and the MBS). This has been discussed in some detail in Sec. 8.3. The CORE server then runs the RSFH heuristic to find a set of alternate routes for the flow in question, which can then be used to route the flow if it is admissible. RSFH uses some simplifications to enable efficient working. For e.g. to compute permissible routes given bandwidth and delay requirements, it just looks at the minimum delay per hop given sufficient bandwidth is available per hop (and this bandwidth can be reserved in a guaranteed manner using the reservation mechanisms provided by reservation based WMNs). This reduces the problem to a simple case of restricting the number of hops taken by a flow to a limited maximum. As in our network we want to provide the best QoS which is possible, infeasible routes will not be considered at all. Further, we will then try to route flows along the shortest-hop paths (given that sufficient bandwidth can be reserved for the flow along such a route, this will ensure the minimum delay). Additionally, based on the QoS restrictions, longer routes can also be considered for the flow. A set of routes can thus be computed for each flow centrally by the CORE server, or each flow can provide the permissible set of routes to the CORE server with the bandwidth and QoS information.

Note that CORE, as its name implies, aims to extend existing and supplement existing routing functionality in the WMN to support the joint optimization of routing, scheduling, and network coding in the WMN. Thus, such information about the feasible set of routes can be provided and maintained by the standard routing algorithm used in the WMN to the CORE server where possible. If this information is not provided, and the CORE server just gets the QoS requirements from the flow at initialization of the flow, then it selects such a set of routes for the flow. QoS information about the flow can also be sent via the periodic demand update messages discussed previously in Sec. 8.3 and presented in detail in Sec. B.5.

A set of routes for a flow can be selected by the CORE server for example, by selecting the shortest-hop routes for the flow and additionally selecting a set of longer routes having more hops up to a preset maximum upper bound on the additional number of hops. Assume that we have thus obtained a set of routes $\mathscr{R}_{f_i}$ which are feasible with respect to the QoS requirements for flow f_i. Now, *RSFH* limits the maximum number of routes which will be considered by the *OptRC* and *MaxSch* heuristics to a maximum of κ routes per flow. In case the set $\mathscr{R}_{f_i}$ consists of a larger number of routes, i.e. $|\mathscr{R}_{f_i}| > \kappa$, then *RSFH* uses a filtering procedure to keep only κ routes which will be considered by the other heuristics. To do this *RSFH* sorts the routes in

Algorithm 1 *RSFH* Algorithm

Definitions:
$\mathscr{F}_{new}$: Set of new flows for admission to the WMN for which no routes have been found.
f_i: flow i from the set $\mathscr{F}_{new}$.
$\mathscr{R}_{f_i}$: Set of feasible (w.r.t. QoS) routes for flow f_i.
κ: The maximum number of routes to be chosen per flow.
$\mathscr{R}_{f_i}^{'}$: Filtered set of routes for flow f_i.
nextlow($\mathscr{R}_{f_i}$): Subroutine which returns the cheapest route from within the set $\mathscr{R}_{f_i}$ in terms of blocking path cost.
remove($\mathscr{R}$, r)): Subroutine which removes the route r from the set of routes $\mathscr{R}$ and returns the remaining set.

1: **for** $i = 1$ to $|\mathscr{F}_{new}|$ **do**
2: $\quad\mathscr{R}_{f_i}^{'} \leftarrow \emptyset$
3: $\quad ctr \leftarrow 0$
4: $\quad$**while** $(ctr \leq \kappa)$ **and** $(\mathscr{R}_{f_i} \neq \emptyset)$ **do**
5: $\quad\quad r \leftarrow nextlow(\mathscr{R}_{f_i})$
6: $\quad\quad \mathscr{R}_{f_i}^{'} \leftarrow \mathscr{R}_{f_i}^{'} \cup r$
7: $\quad\quad \mathscr{R}_{f_i} \leftarrow remove(\mathscr{R}_{f_i}, r)$
8: $\quad\quad ctr \leftarrow ctr + 1$
9: $\quad$**end while**
10: **end for**

the set $\mathscr{R}_{f_i}$ in ascending order of the blocking costs of the individual routes $\varsigma^{sd}(\mathscr{P}_s^d)$ as defined in Eq. (8.3). From this sorted list *RSFH* picks up at most the first κ routes as candidate routes which will be considered by the heuristics *OptRC* and *MaxSch*.

The goal hereby is to reduce the computational effort needed for the remaining heuristics by keeping an upper bound on the solution space. Furthermore, we want to jointly optimize the routing, scheduling, and use network coding in the WMN in order to enhance the traffic carrying capacity of the WMN. By restricting the solutions computed by CORE's heuristics to routes with the minimum blocking cost, we ensure that more spatial reuse of bandwidth is possible. The pseudocode for *RSFH* is shown in Alg. 1. The set of routes thus selected (a maximum of κ routes per flow) then are used as an input for the *OptRC* heuristic. The set of feasible routes $\mathscr{R}_{f_i}$ can be, as previously discussed, either provided directly by the standard routing algorithm being run in the WMN, or it can be computed by *RSFH* using a standard k-shortest path (see for example [33]) algorithm.

8.4.2 Optimal Route Combination Heuristic (*OptRC*)

The solution to the *OptRC* problem uses the solution to the *MaxSch* problem (see 8.4.3) as a subroutine (within the procedure $combine()$ shown in the pseudocode for Algorithm 2). The *OptRC* problem can be formulated as follows. "*Given a set of flows, their traffic demands, and a set of routes per flow, what is the optimal combination of routes (by choosing one route per flow) such that: the maximum traffic demand can be supported in the network, the overall interference in the network is minimized when using the schedule produced by the MaxSch subroutine, and bandwidth savings via network coding are maximized?*"

A subset of the flows from the given set of flows may be dropped if they cannot be scheduled using the *MaxSch* routine or if the network capacity is insufficient, i.e. only an insufficient number of minislots is available. In particular, the solution to the *OptRC* problem is a set of routes which contains a single route for each source-destination pair. Algorithm 2 shows our proposed algorithm as pseudocode. Instead of finding an optimum route set for all flows at once, the algorithm operates stepwise. We next define how Algorithm 2 divides the *OptRC* problem into smaller optimization steps.

Algorithm 2 *OptRC* Algorithm

Definitions:

f: List of flows to be processed, ordered according to the flow's importance.
f_0: most important, unprocessed flow (i.e. first element of f).
a: Set of flows for which a route has already been found.
p: A partition, i.e. a self sorting list of flows; ordered according to the flow's importance.
P: Self sorting list of partitions (i.e. list of lists of flows) with elements P_i; ordered by the importance of the first flow of the elements.
P_0: First element of P, i.e. the partition containing the most important, unprocessed flow.

1: $a \leftarrow \emptyset$
2: **repeat**
3: $p \leftarrow \emptyset$
4: $P \leftarrow \emptyset$
5: $comb \leftarrow 1$
6: **while** $comb \cdot k_0 \leq \Delta$ **do** ▷ Add flows to p until $comb > \Delta$
7: $comb \leftarrow comb \cdot k_0$
8: $p \leftarrow p \cup \{f_0\}$
9: $f \leftarrow f \setminus \{f_0\}$
10: **end while**
11:
12: $P \leftarrow P \cup p$
13: **repeat** ▷ Binary search
14: $best \leftarrow \text{combine}(P_0 \cup a)$ ▷ Search best set of routes
15: **if** $best = \emptyset$ **then** ▷ No valid combination found
16: **if** $|P_0| = 1$ **then** ▷ P_0 contains only one element
17: $P \leftarrow P \setminus \{P_0\}$ ▷ Flow in P_0 not routeable
18: **else** ▷ Split current partition in two partitions
19: $np \leftarrow \left\{ P_{0,x} : x \geq \left\lfloor \frac{|P_0|}{2} \right\rfloor \right\}$
20: $P_0 \leftarrow P_0 \setminus \{t : t \in np\}$
21: $P \leftarrow P \cup \{np\}$ ▷ np has second position in P
22: **end if**
23: **else**
24: $a \leftarrow a \cup P_0$
25: $P \leftarrow P \setminus \{P_0\}$
26: **end if**
27: **until** $P = \emptyset$
28: fixRoutes($best$) ▷ Fix the flows to the routes currently found
29: **until** $f = \emptyset$

From the list of yet to be routed (non-processed) flows f, t flows are chosen, such that

$$\prod_{s=1}^{t} k_s \leq \Delta \tag{8.17}$$

where Δ is a constant set by the network operator and k_i is the number of possible routes of the ith non-processed flow (Lines 3–11). These t flows form one partition p as shown in Algorithm 2. In our algorithm we add flows to a partition till the bound given by Eq. (8.17) is reached. We always select the first flow (f_0) in the list of flows f as the next flow to be added to a partition of flows to be optimized in parallel. Hence, the order of flows in f plays an important role in the performance of Algorithm 2.

For the results in this thesis (unless otherwise specified, e.g. for Sec. 9.3), we sorted the flows based on their traffic demand and they are listed in f such that a flow which has a higher traffic demand is placed before flows with lower traffic demands. Optimizing a larger number of flows in parallel is beneficial, because it enables the computation of a schedule, which allows maximal concurrent transmissions for the considered flows, by adapting the routes for all these flows. Thus, the main goal of Algorithm 2 is to efficiently search for a combination of routes which allows the maximum traffic demand to be supported. Algorithm 2 uses binary search within a given partition to quickly find the subpartition (P_0) of flows which can be jointly scheduled when the given partition of t flows cannot be jointly scheduled (see lines 13–27).

The procedure *combine*() passes all possible route combinations for the given flows as argument to the *MaxSch* algorithm and returns the best schedulable set of routes. The binary search enables us at the same time to find out flows which cannot be scheduled by the *MaxSch* routine together with the flows which have been already scheduled and for which the routes have already been fixed. Flows for which no schedulable solution could be found are excluded from any further calculations. All other flows regarded in this step are fixed to the route that has just been found. This procedure repeats until all flows have been considered.

In the case that each of the t flows has κ routing possibilities, the maximum number of route combinations $\hat{C}(\kappa, t)$ that have to be checked (implemented using the *combine*() procedure which calls the computational expensive *MaxSch* algorithm for each possible route combination) until all t flows have been processed can be calculated iteratively by

$$\begin{aligned}
\hat{C}(\kappa, 0) &= 0 \\
\hat{C}(\kappa, 1) &= \kappa \\
\hat{C}(\kappa, 2) &= \kappa^2 + 2\kappa \\
\hat{C}(\kappa, 3) &= \kappa^3 + \kappa^2 + 3\kappa \\
\hat{C}(\kappa, 4) &= \kappa^4 + 2\kappa^2 + 4\kappa \\
&\dots \\
\hat{C}(\kappa, t) &= \kappa^t + \hat{C}(\kappa, \lceil t/2 \rceil) + \hat{C}(\kappa, \lfloor t/2 \rfloor). \qquad (8.18)
\end{aligned}$$

One can see from Eq. (8.18) that the number of possible route combinations and, thus, the number of schedules that have to be calculated, increases exponentially. The operator can choose between processing either more flows in parallel or allowing more routing alternatives for each flow.

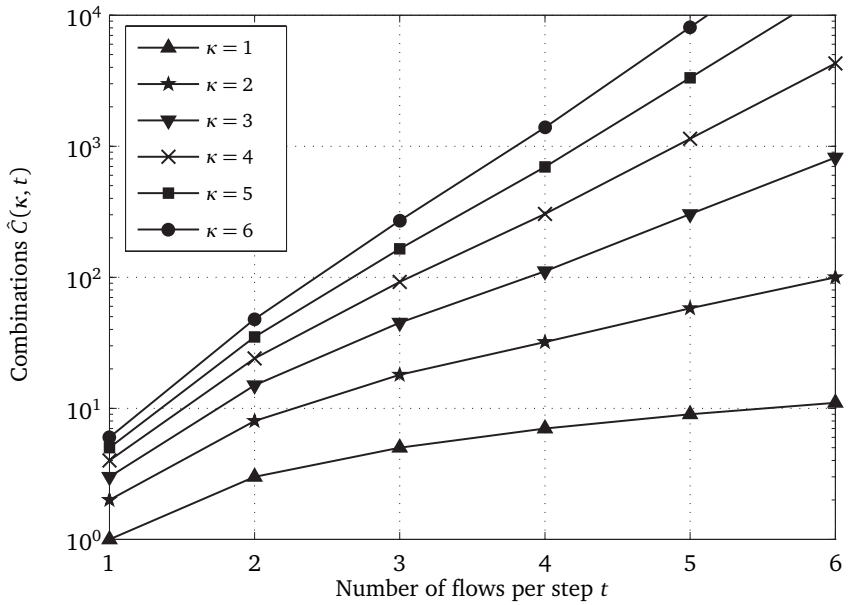


Figure 8.5: Number of Possible Route Combinations $\hat{C}(\kappa, t)$ for Varying κ and t

Fig. 8.5 shows the trends for the number of possible route combinations as given by Eq. (8.18).

8.4.3 Maximal Scheduling Heuristic (*MaxSch*)

The *MaxSch* heuristic is given a set of flows and one route per flow. It first computes the bandwidth required (demand) per link in the network for the given set of flows and routes. The *MaxSch* algorithm next attempts to find a maximal schedule for the demand, returning the amount of (minislot,link) tuples blocked by the computed schedule. The returned value can, in general, be any metric which allows the calling function to evaluate the quality of the computed schedule. In case the given set of flows cannot be scheduled due to bandwidth limitations, an error value is returned. The *MaxSch* subroutine is called repeatedly by the $combine()$ procedure shown in Algorithm 2, to find the best (minimum blocking) route combination.

For our subproblem, a maximal schedule is a set of scheduled data transmissions (for the given traffic demand) per minislot such that no further non-conflicting data transmissions can be scheduled. The above definition is similar to the definition for the maximal slot assignment presented in [17]. It has been shown previously that maximum system capacity can be attained by using a throughput-optimal scheduling algorithm where at each time slot, the schedule is computed such that the queue-weighted rate-sum is maximized (see Refs. [113, 88, 63, 62] for details). Further, in literature, it has been shown that greedy maximal scheduling can be viewed as an approximation to such throughput optimal algorithms. Hence, we use such a maximal scheduling approach to compute centrally the possibility of accommodating the given flows with their respective bandwidth demands under the assumption of near-optimal scheduling. This gives us an upper bound to the number of flows with the respective demands which can be scheduled along the chosen routes in the WMN without huge queueing delays.

To find a maximal schedule we use an adapted version of the greedy maximal scheduling algorithm presented in [62] (similar scheduling algorithms may also be found in [17, 114]). The *MaxSch* heuristic assigns the first minislot to the link with the highest demand. Thereby, the corresponding set of conflicting links cannot be activated in the same minislot. Of the remaining links, again the link with highest demand is chosen and assigned to the current minislot. The procedure repeats until no more links can be activated in this minislot. The demand of all active links is then reduced by one and the heuristic restarts with the updated demands per link for the next minislot.

We adapted the *MaxSch* heuristic described above so that it can also be used for scheduling network coding transmissions (which are multicast transmissions in contrast to the normal unicast transmissions in a WMN). In the presence of per-link encryption in the WMN (if present), network coding via opportunistic listening as outlined in [49] is not possible. Hence, network coding in such a scenario is only possible when we have a traffic setup similar to the "Alice-Relay-Bob" scenario outlined in Ref. [49]. We do not consider the central optimization of SONC opportunities which require overhearing, this is because of the additional overhead involved in bandwidth reservation, as well as control traffic. Furthermore, it is more difficult to coordinate and set up a session using overhearing as compared to a scenario like the "Alice-Relay-Bob".

We use the flow and routing information to determine all possible network coding opportunities and create a virtual network coding link for each of them. The demand on these virtual network coding links is equal to the minimum of the demands of the corresponding two unicast transmissions, which are now replaced by transmissions on the virtual network coding link. Since some of the demand is now served by the network coding links, the demand on the corresponding unicast links is reduced by the same amount. However, as a single network coding transmission can transport the double amount of data compared to a normal transmission, network coding links should be preferred by the MaxSch heuristic. Consequently, when choosing a link to schedule next, we consider the effective demand for network coding links to be twice the real demand (in minislots).

Algorithm 3 *MaxSch* Algorithm

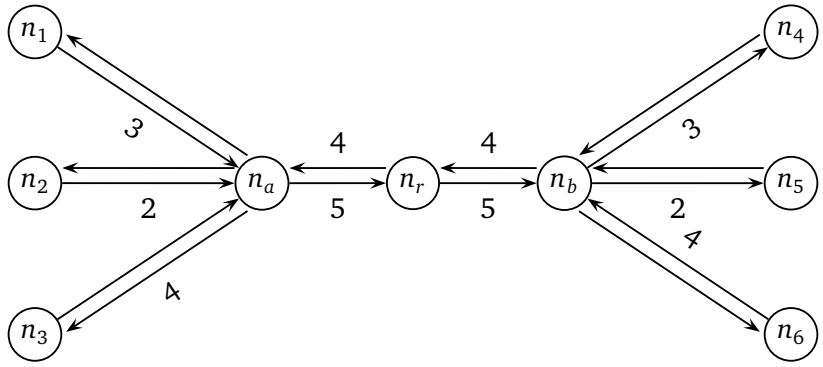
Definitions:
ERROR: Value indicating that no schedule satisfying the demand of all the flows could be found.
$\mathscr{F}$: Set of flows labelled from f_1 to $f_{|\mathscr{F}|}$.
$\mathscr{R}$: Set of routes with one route r_i corresponding to each flow f_i from the set $\mathscr{F}$.
M: Maximum number of minislots to be used for schedule.
G: Graph $(\mathscr{V},\mathscr{E})$ for the WMN.
$\mathscr{V}$: Set of vertices (nodes) in the WMN.
$\mathscr{E}$: Set of directed edges (links) in the WMN.
$\mathscr{E}'$: Augmented set of edges including virtual multicast links.
$D(e)$: Total demand in minislots per frame for edge $e \in \mathscr{E}'$.
$UD(e)$: Unsatisfied demand in minislots per frame for edge $e \in \mathscr{E}'$.
$\omega(e)$: Weight giving preference to demand of an edge $e \in \mathscr{E}'$.

$$\omega(e) = \begin{cases} 1, & \text{if } e \in \mathscr{E} \\ 2, & \text{if } e \in \{\mathscr{E}' \setminus \mathscr{E}\} \end{cases}$$

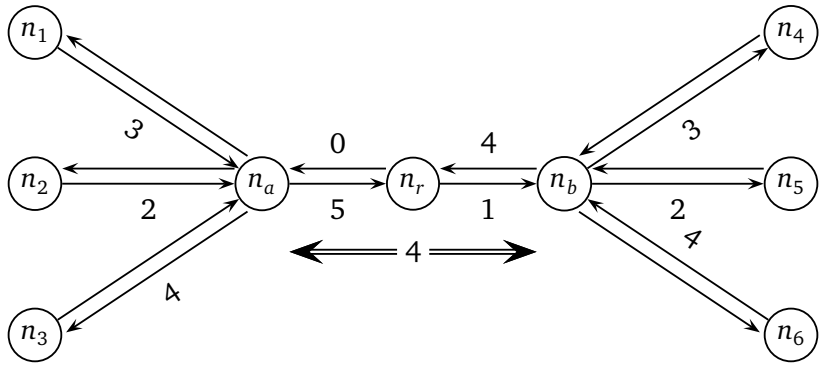
```
1:  procedure MaxSch(F,R,M, G)
2:      ctr ← 0
3:      m ← 0
4:      E' ← computeDemandPerLinkWithNC(F,R)        ▷ Compute NC virtual links and demand on each individual link
5:      while (m < M) do
6:          blocked ← ∅
7:          active ← ∅
8:          em ← arg max_{e∈E'} ω(e) × UD(e)
9:          if (UD(em) = 0) then
10:             return ctr
11:         end if
12:         allocate(m,em)                          ▷ Allocate slot m to link em
13:         active ← (active ∪ em)
14:         blocked ← (blocked ∪ blockedBy(em))     ▷ blockedBy(e) returns set of links which cannot be activated in parallel to e
15:         T ← E' \ em
16:         while T ≠ ∅ do
17:             em ← arg max_{e∈T} ω(e) × UD(e)
18:             if (em ∉ blocked) then
19:                 allocate(m,em)
20:                 active ← (active ∪ em)
21:                 blocked ← (blocked ∪ blockedBy(em))
22:             end if
23:             T ← T \ em
24:         end while
25:         m ← (m + 1)
26:         ctr ← (ctr + |blocked|)
27:     end while
28:     em ← arg max_{e∈E'} ω(e) × UD(e)
29:     if (UD(em) = 0) then
30:         return ctr
31:     else
32:         return ERROR
33:     end if
34: end procedure
```

Algorithm 3 shows the *MaxSch* algorithm as pseudocode in detail. *MaxSch* takes as an input a model of the WMN given as a directed graph G with the links in G corresponding to the links in the WMN. It also gets the set of flows $\mathscr{F}$, where a single route r_i is associated with each flow f_i. Given the flows (which provides also information about their individual demand), and the set of routes the *MaxSch* algorithm uses the function *computeDemandPerLinkWithNC()* as shown in Algo. 3 line 4. Fig. 8.6 visualizes the the actions carried out by the above function.

Consider the WMN topology shown in Fig. 8.6. Say we have three flows in the WMN, one from n_1 to n_4 with a bandwidth demand of 3 minislots per frame. Let us assume additional flows in the network $f(n_2, n_5)$ and $f(n_6, n_3)$ with bandwidth requirements of 2, and 4 minislots



(a) Per-Link Demand Without Network Coding



(b) Per-Link Demand With Network Coding

Figure 8.6: Visualization of Computation of Per-Link Bandwidth Demand for the Wireless Mesh Network (a) Without Network Coding and (b) With Network Coding

per frame respectively. For the given WMN topology there are only unique paths possible for the three individual flows. Assuming these routes Fig. 8.6(a) shows the number of minislots needed to be reserved for the individual links in the WMN along which packets will be routed. Where no number is shown against a link a demand of 0 minislots exists, and thus no slots need to be reserved for that link. This computation is done as a first step in the function *computeDemand-PerLinkWithNC()*. The function then identifies at node n_r an opportunity for network coding, where packets belonging to flows $f(n_1, n_4)$ and $f(n_2, n_5)$ can be coded with packets from flow $f(n_6, n_3)$ in the "Alice–Relay–Bob" constellation we discussed earlier.

From the example it is clear that the total bandwidth demand in one direction across the relay i.e. from $n_a \rightarrow n_b$ is 5 minislots and in the other direction i.e. from $n_b \rightarrow n_a$ on an average we

will have a traffic flow requiring 4 minislots per frame. Thus, to enable suitable and efficient network coding, without delays waiting for packets belonging to one of the local streams, it is meaningful to code traffic amounting to 4 minislots per frame from each of the streams together. Thus, CORE here assumes that network coding would be enabled at node n_r for traffic requiring a reservation of up to 4 minislots. Thus, the function *computeDemandPerLinkWithNC()* will now generate a virtual network coding link (multicast link) as an outgoing link at node n_r transporting packets to nodes n_a and n_b. This virtual multicast link is shown using double lined arrows in Fig. 8.6(b). The function *computeDemandPerLinkWithNC()* will then associate a traffic demand of 4 minislots per frame with this network coding link using the justification as discussed previously.

However, as a part of the traffic is now being transported as network coded traffic, the normal unicast traffic on the corresponding links in the WMN will not require the same number of minislots per frame as were required without network coding in the WMN. The function *computeDemandPerLinkWithNC()* reduces these unicast traffic demands on the affected links to adjust for the traffic forwarded via network coding. This will then lead to the traffic demands as shown in Fig. 8.6(b).

Thus from the above computation, we get the demand per link for each link in the set $\mathscr{E}'$ which, is basically the set of links $\mathscr{E}$ with the additional virtual links computed by the function *computeDemandPerLinkWithNC()*. The *MaxSch* algorithm then iterates through all the available minislots for scheduling. At each iteration it first allocates the slot to the link $e \in \mathscr{E}'$ having the maximum unsatisfied demand. The same slot is then allocated to all other links with unsatisfied demands in descending order of their unsatisfied demands. When allocating the slot to a link, care is taken to see that the link is not blocked from transmission in that slot by the already scheduled transmissions. These operations are shown in Algo. 3 lines: 5–27.

When, after iterating through all the available slots for scheduling, *MaxSch* notices that there are still links in $\mathscr{E}'$ with an unsatisfied demand, it is clear that the maximal schedule found does not support the required traffic demands of the flows. This is shown in Algo. 3 lines: 28–33. In case the schedule found does not satisfy the requirements an ERROR value is returned indicating the failure to find a schedule. If, on the other hand, a schedule could indeed be found, *MaxSch* returns the number of (minislot,link) tuples which were blocked by the maximal schedule. This is a measure of the goodness of the scheduling under the constraint of the given routes and traffic demands for the flows. The lower the value the more the space available in the WMN to accommodate additional traffic. However, as discussed previously, *MaxSch* can return any other metric of goodness which allows the comparison of two maximal schedules. One such metric could be the number of transmissions (both unicast and multicast) which could be scheduled with suitable weight added for the multicast transmissions to reflect the fact that more information is transported by the multicast network coded transmissions we are referring to in our context in comparison to standard unicast transmissions. This value of goodness of the routes chosen for the given flows can then be used by the procedures which use *MaxSch* as a subroutine (e.g. *combine()* in *OptRC*).

Having looked at the heuristics run at the CORE server in depth, in the next section, we will look at the mechanisms we propose in the CORE framework so that the centrally computed solution can be efficiently deployed in the WMN using the distributed components present at the individual nodes in the WMN.

8.5 Phase III: Mechanisms for Deployment of the Centrally Computed Solution

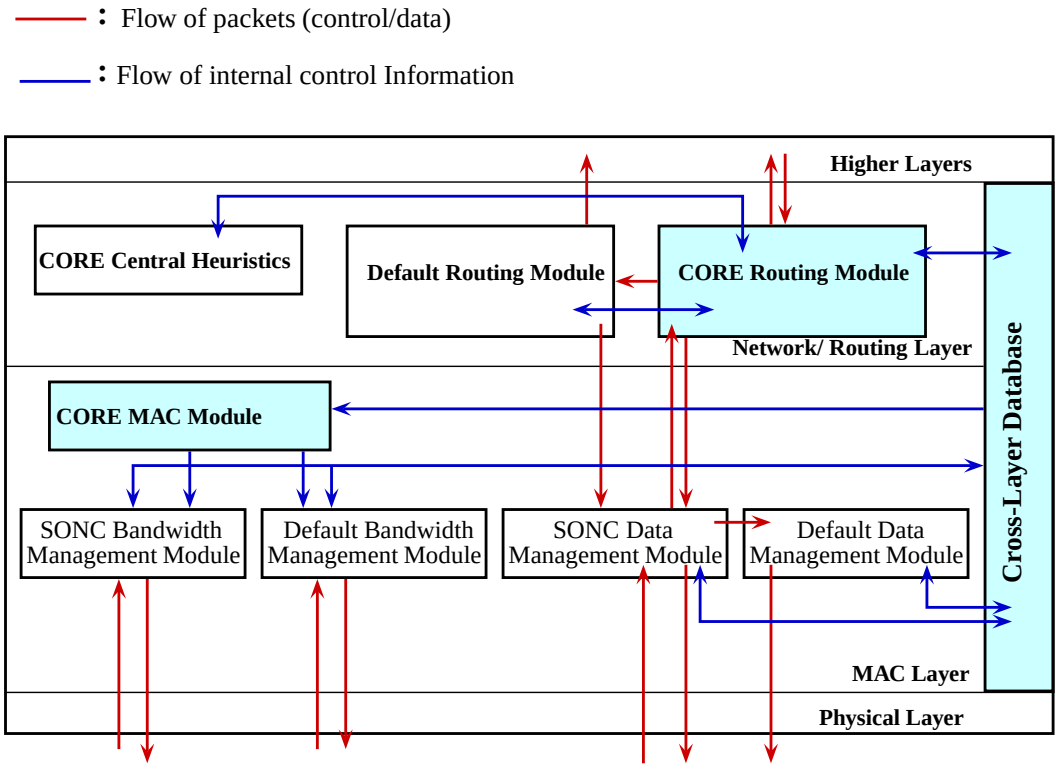
From Sec. 8.3, Sec. 8.4 and the overview in Sec. 7.2 we have seen that though CORE computes its optimization solution centrally at the CORE server, it uses components at each node in the WMN to collect information about the flows in the network (Sec. 8.3) as well as for deploying the centrally computed solution. In this section we discuss the latter in some detail. CORE uses its central heuristics to seek a route combination such that we have a single route per flow and there exists a maximal schedule (as computed by the *MaxSch* algorithm) which permits the flows to be routed without huge packet backlogs when routed along the chosen routes. The routes are chosen such that the overall interference in the WMN (as given by Eq. (8.3)) is minimum.

CORE uses distributed components to collect information about the flows as well as to deploy the computed solution in order to keep the control overhead to a minimum. If the entire schedule would be centrally computed throughout the WMN as well as communicated to the nodes throughout the WMN this would increase the control traffic overhead. In addition the schedule would need to be centrally computed and distributed for each frame as otherwise there would be no means to accommodate traffic bursts at the individual nodes in the WMN. This is in addition to the corresponding routing changes which, would also be needed more frequently, and the routing updates which would be needed to be sent more often. This would make such a solution infeasible except for very small WMN sizes. Additionally, as we have seen in Part II of this thesis, in reservation based WMNs, network coding can only be efficiently deployed by making coding decisions for packet streams and not on a packet-by-packet basis.

To avoid this, in the CORE framework, nodes report only the mean traffic demand over a long (as defined by the parameter for the average, see Eq. 8.15) time frame. The schedule is computed considering this long-term demand for the reported traffic flows, and the routing changes (if any) along with the corresponding bandwidth reservation changes are communicated to the nodes in the WMN. However, traffic in the WMN is by no means stable at the mean data rate, and hence it is not possible to ignore the short-term bandwidth demand fluctuations completely. To tackle this issue the CORE framework proposes an intelligent division of responsibilities and uses a conservative approach when computing the central solution, and the maximal schedule. We will describe these details and the justification for the deployment choices made when designing the CORE framework in detail in this section-

Fig. 8.7 shows the main components of the CORE framework as in Fig. 8.3. However, in Fig. 8.7 the components of the CORE framework which are vital during the deployment phase (Phase III) of CORE's centrally computed solution are shown shaded. We see from Fig. 8.7, that the **CORE Routing Module**, the **CORE MAC Module**, and the **Cross-Layer Database** play a vital role in deploying the solution computed by CORE in the WMN.

Referring to the discussion in Sec. 8.4 we see that the major task of the heuristics was to find a combination of routes — one route per flow — which would enhance the number of network coding opportunities in the network, as well as permit a maximal schedule which would block the minimum number of (minislot, link) tuples in the WMN, thereby leaving room for more additional traffic to be admitted to the WMN. Thus, one sees that CORE needs to control and modify the routes taken by the flows, as well as to some extent the transmission schedules at the nodes in order to implement the computed solution. The **CORE Routing Module** is responsible for allowing CORE to modify the routes to correspond to the centrally computed route combination. The **CORE MAC Module** is responsible for adapting the transmission schedules to match the route changes. The **Cross-Layer Database** permits data to be efficiently shared between the modules in the different layers to allow better interworking of the individual components. We



Protocol Stack of the Nodes in the WMN showing Vital CORE Components*

(*Note: vital components in the phase III heuristics in CORE are shown shaded)

Figure 8.7: CORE's Architecture Showing the Key Phase III Components

next explain in detail the role and functions of each of the above modules in detail followed by a discussion of how they fit together as a whole in the CORE framework.

8.5.1 CORE Routing Module

The **CORE Routing Module** is responsible for ensuring that the packets for individual flows in the WMN are routed along paths chosen to be the most promising routing combination by the CORE Central Server. When the CORE Server has finished computing its route combination for the individual flows in the network, it has in effect for each intermediate node in the WMN a next-hop for the respective flow. As discussed previously we identify a traffic flow for this thesis as a stream of packets from a given source node to a given destination node in the WMN and is denoted using the notation $f(s,d)$ where s is the source node, and d is the destination node for the packets belonging to the traffic flow. For routing purposes, source nodes or destination nodes for flows which lie outside the WMN are replaced by the ingress/egress nodes respectively. In our discussion we have assumed, without loss of generality, this ingress/egress node to be the MBS.

Thus, we can say that at an intermediate node n_j, the next-hop of the packet belonging to a particular flow f_i does not depend on the final destination of the packet (as is the case when default shortest-path routing is used in the WMN), but depends on the flow to which the packet belongs. To state concisely, we have the routing function *CR()* as shown in Eq. (8.19) and Eq. (8.20) at each node for routing packets.

$$CR : \mathscr{F} \times \mathscr{V} \rightarrow \mathscr{V} \tag{8.19}$$

$$CR : \mathscr{V} \times \mathscr{V} \times \mathscr{V} \rightarrow \mathscr{V} \tag{8.20}$$

CR is thus a mapping from $\mathscr{F} \times \mathscr{V}$ to $\mathscr{V}$, where $\mathscr{F}$ is the set of flows, and $\mathscr{V}$ is the set of vertices or nodes in G=($\mathscr{V}$,$\mathscr{E}$). Thus, for a flow $f \in \mathscr{F}$ and at a node $v \in \mathscr{V}$ there is a chosen next-hop $nh \in \mathscr{V}$. Additional logical constraints do exist, e.g. $nh \in Nbr(v)$. However, given that we define a flow $f(s,d)$ as the stream of packets from a source to a given destination, we can also define the routing function *CR* as given by Eq. (8.20) where $\mathscr{F}$ is replaced by $\mathscr{V} \times \mathscr{V}$ which gives the tuple of (source, destination) pairs in the WMN. For our concrete implementation we use the version in Eq. (8.20) whereby on receiving a packet a node will determine it's next-hop based on the source of the packet, and the destination node of the packet. The third element of the tuple in the domain of *CR*, i.e. the node doing the forwarding itself is then implicitly chosen to be itself at each individual node in the WMN. Note that this implies additional overhead for maintaining the routing tables whereby we see that we now need routing tables with size $O(|\mathscr{V}|^3)$ instead of $O(|\mathscr{V}|^2)$ when the default routing is used. However, one should note that these are worst-case asymptotic space complexity estimates, and in practice the additional space required at any point in time will be limited. This is as a node v need only maintain the additional flow-based routing information only for flows which are currently being handled (routed) by the node. In the general case in a WMN it is highly unlikely that a given node will be responsible for routing flows from all other nodes to all other nodes in the WMN.

Thus, the **CORE Routing Module** shown in Fig. 8.7 basically extends the routing functionality provided by the **Default Routing Module** present at each node in the WMN. To understand this we will look at the flow of data and control packets in a CORE enabled node shortly. As shown in Fig. 8.7, packets arriving from the higher layers at the network/routing layer are first inspected by the **CORE Routing Module**. If the **CORE Routing Module** has routing information which permits it to determine the next-hop for the packet then it will mark the packet appropriately

and hand it over to the lower layer for transmission. On the other hand, if the **CORE Routing Module** does not have routing information for the packet at hand it will hand over the packet to the **Default Routing Module** for further processing, which, after looking up its own routing table will then choose the appropriate next-hop and hand the packet to the lower layer for transmission to the selected next-hop.

Similarly, packets received from the medium (PHY layer) at a node will be handed over to the MAC layer. If the packet at hand is not a packet for processing at the MAC layer, it will be handed to the **CORE Routing Module**. The **CORE Routing Module**, similar to packets it receives from the upper layers, will inspect the packet to either determine the next-hop, or hand it to the upper layer if the packet has reached its final destination. If the **CORE Routing Module** cannot process the packet correctly for lack of information it will hand over the packet to the **Default Routing Module** at a node for further processing. Thus, from the above discussion we see that the **CORE Routing Module** provides an extended routing functionality to a node (hence the term "Routing Extensions" in CORE). It permits routing decisions to be taken per packet not only based on the final destination of a packet but also dependent on the flow to which a packet belongs. Besides the above, the **CORE Routing Module** is also responsible for the tasks as discussed in Sec. 8.2.2.

In our concrete implementation of the CORE framework, we used the CORE-RTUPDT (CORE routing update) as discussed in Sec. B.5 to enable the CORE Server to notify the individual nodes in the WMN of routing changes they should incorporate at the respective node in the WMN. The routing changes are activated by all the nodes in the WMN simultaneously at the frame specified by the CORE-RTUPDT message. Further details of the actual implementation can be found in Sec. B.5. When the control messages related to CORE are received by the **CORE Routing Module** either from the lower layer or from the **CORE Central Heuristics** module, these are processed by the **CORE Routing Module** to take appropriate action (i.e. make routing changes or send CORE-RTUPDT messages (only at the CORE Server)). As a result of receiving and processing these CORE control messages the **CORE Routing Module** also updates certain information in the **Cross-Layer Database**, e.g. the estimated demand for a particular flow, and thereby the estimated long-term demand for a given link. In addition the frame in which the new routing changes are to be activated is also stored in the **Cross-Layer Database**. This information is used by the **CORE MAC Module** to make appropriate bandwidth reservations and adapt existing bandwidth reservations where needed.

In the next section we will look at the **CORE MAC Module** and its functions in more detail.

8.5.2 CORE MAC Module

As discussed earlier, the CORE Server computes locally a permissible maximal schedule using the *MaxSch* algorithm (Algo. (3)). However, though it is possible to use means to deploy the centrally computed schedule throughout the WMN (see for e.g. Ref. [72] and Ref. [1]), this leads to traffic overhead. In addition, it is not possible to schedule every burst of traffic in the WMN using such a centrally computed schedule. This would involve a lot of overhead, firstly to keep the central server constantly informed of the number of packets (bytes) in the queues at each individual link in the WMN at every node in each frame so that the CORE Server can compute a complete transmission schedule for the current up-to-date demand at each node in the WMN.

Hence, in the CORE framework we proposed that the CORE Server should optimize the routes and schedule only by looking at the long-term demand of the flows ((8.15)). Thus, the CORE Server is not informed about each traffic burst, and only optimizes the route combination and

computes a maximal schedule using the long-term demands of the flows. Traffic bursts exist in the WMN though. To allow for such bursts to be accommodated in the WMN we use a conservative approach at the CORE Server when computing the permissible maximal schedules. When computing the maximal schedule using the *MaxSch* algorithm, we propose that the algorithm use as parameter M only a fraction of the minislots actually available in the WMN for distributed scheduling. For the work presented here and for our implementation we set this value conservatively to seventy percent[§]. Thus, when computing the maximal schedule the CORE Server will assume that only seventy percent of the actually available minislots are present for the sake of computing its schedule and checking for the feasibility of the schedule. This design has two underlying justifications, for one, the maximal schedule computed centrally uses global knowledge which is not available at the individual nodes in the WMN, also even when such global knowledge would be present reaching the maximal schedule using distributed means would require a lot of coordination overhead, especially when the schedule for short-term traffic bursts is not accounted for by the maximal schedule. Else we would also need to schedule and account for the traffic bursts in the maximal schedule which, in practice, is quite infeasible considering the overhead which would be involved. Secondly, we need to have free room available to accommodate the short term bursts of traffic in the WMN.

Hence, besides using only a fraction of the total minislots available for computing the maximal schedule at the CORE Server, we also propose a mechanism to control the bandwidth reservation permitted at the nodes in the WMN when the CORE framework is active. We have seen in Chap. 3 that the MeSH mode offers different persistences for bandwidth reservation. Bandwidth once reserved with persistences which are limited in the number of frames for which the bandwidth will be allocated (i.e. $Per_1, \ldots, Per_{128}$) cannot be freed to enable use by other nodes, when not required for the link to which the bandwidth has been allocated. Thus, such slots will only be available again when the reservation duration has expired (i.e. the frames for which the bandwidth has been allocated has passed). On the other hand it is possible to reserve bandwidth for a link such that once allocated it remains allocated to the link unless explicitly freed (i.e. using Per_∞). In this case, however, it is possible to free the slots thus reserved by initiating a cancel three-way handshake (see [83, 42] for details). We propose that when the CORE framework is active in the WMN, the Default Bandwidth Management Module (shown in Fig. 8.7) be restricted from reserving bandwidth with persistence Per_∞. Only the **CORE MAC Module** may initiate bandwidth reservations with persistence Per_∞ and their cancellations. The Default Bandwidth Management Module is then restricted to bandwidth reservations with persistences Per_1, ..., Per_{128}.

The above design choice has two aims:

1. As the CORE Server computes the route combination based on long-term demand of flows, the total long-term demand for a link in the WMN can be simply approximated as the sum of the long-term demands of the individual flows routed along the link by the CORE Server. This, demand is available in the **Cross-Layer Database**, and can be used by the **CORE MAC Module** to initiate long-term (Per_∞) bandwidth reservations on individual links in the WMN. This also permits changing and adapting such reservations when the CORE Server initiates routing changes. This would not be possible if the bandwidth would be reserved via the persistences Per_1, ..., Per_{128}. Furthermore, as the long-term demand is less subject to fluctuations than the instantaneous bandwidth demand it is meaningful to reserve for such demand a stable number of slots per frame over a larger number of frames.
2. In the ideal case the long-term bandwidth demand for each link will be met by the former reservations discussed above. To enable the WMN to also accommodate short-term bursts

§ The parameter allows the network provider to tune the system to his needs.

of traffic, we permit the Default Bandwidth Management Module to reserve bandwidth for such short term demands using the persistences Per_1, ..., Per_{128}. These reservations are short-term reservations, and permit individual nodes in the WMN to additionally reserve bandwidth on individual links for a short duration to tackle short bursts in the arrival of packets for transmission at a node. Additionally, as the Default Bandwidth Management Module carries out only short-term reservations, it is likely that in the long term (i.e. after the current short-term reservations have expired), the individual nodes in the WMN will have sufficient slots free to reserve long-term bandwidth as directed by the route combination specified by the CORE Server.

In general the arbitrated distributed schedule will not match the centrally computed maximal schedule. To allow for some slack in the actual schedule reached, the CORE Server does not use the entire range of slots when computing the maximal schedule. This also leaves some minislots free when the long-term demands as given by the CORE Server specified route combination is deployed. These, in turn, can be used to accommodate short-term traffic bursts.

The information about when the route changes are going to be activated, as well as the new long-term bandwidth demands per outgoing link is stored in the **Cross-Layer Database**. Now, at each control transmission opportunity of the node, the **CORE MAC Module**, looks up the **Cross-Layer Database** to find out if it should issue in advance requests at that transmission opportunity to ensure that bandwidth is available when needed (i.e. at the time the routing changes are activated). Ideally, the entire bandwidth arbitration handshake should be completed before the routing changes are actually activated. Hence, assuming that the routing changes are to be activated in say frame f_a the requests should be issued latest at frame f_r where $(f_a - f_r) \geq (k \times H)$. Where, k is a constant such that $k \geq 1$ and H is the current estimate for the three-way handshake delay. Thus, the **CORE MAC Module** can try to ensure that the required bandwidth will be available at the time the routes are adapted, and the packets do not face additional delays till bandwidth for transmitting these is reserved on all the links along the new path. To indicate to the receiver of a $MSH-DSCH\_Request\_IE$ to know at which frame (or starting at which frame) the requested slots should be granted, we extended the $MSH-DSCH\_Request\_IE$ proposed in the IEEE 802.16-2004 standard [42] by a field which we term *wished-start-frame*. This additional field specifies the frame starting at which the requester would best have a grant for the requested bandwidth. This, enables a much finer control of the bandwidth request process and avoids wastage of granted slots (see [84, 20] for a detailed study of this aspect). However, we note that this is not the only means to achieve the above effect, and it is also possible to achieve the same effect using the IEEE 802.16-2004 standard specifications without any extensions, however, sacrificing efficiency and with much more complexity.

8.5.3 Cross-Layer Database

As mentioned earlier in this section, the **Cross-Layer Database** is one of the components which play an important role in helping deploy the solution computed centrally by the CORE Server. It however, plays a passive role, and is only present to allow for efficient sharing of data across components at different layers in the protocol stack in the WMN. Thus, in principle, there are other means to implement the functionality provided by the **Cross-Layer Database**.

As mentioned in Ref. [106], the addition of a **Cross-Layer Database** can be seen to be similar to the concept of introducing an additional layer in the protocol stack. In Fig. 8.7 we show the **Cross-Layer Database** as spanning the MAC and Networking layers. This is because of the fact that for the purposes of the investigations in the scope of this thesis, an interaction among these two layers is of primary importance. On the other hand, implementers of the CORE Framework are free to extend the functionality of the **Cross-Layer Database** to span further layers.

However, as mentioned in Ref. [50], caution should be exercised when breaking with the compartmentalized development environment offered by the strictly layered protocol architectures. In our design the **Cross-Layer Database** is primarily responsible for storing information about the estimated bandwidth demands (both long-term and short-term) for each outgoing link at the node in the WMN. These measurements will usually be made and kept at the MAC layer, especially in reservation based WMNs to allow efficient bandwidth reservation. Additional to these, we maintain at each node in the WMN the long-term data rate estimates for flows originating at the node in the WMN. This information, if needed, can also be used at the MAC layer for more sophisticated bandwidth reservation schemes (e.g. Ref. [84]). The **CORE Routing Module** stores here the measured (estimated) long-term data rate for each flow originating at the node, this can also be done at the MAC layer though. It also stores for each flow originating at the node the last reported long-term data rate, and also the corresponding frame number in which the long-term demand for the flow was reported to the CORE Server. This plays a vital role in Phase I of CORE (see explanation in Sec. 8.3.2 and Eq. (8.16)).

Additional to storing the measured long-term data rate for flows, the **CORE Routing Module** also stores the expected future long-term bandwidth demand per outgoing link based on the routing update messages it receives from the CORE Server. Further, it stores also the frame number at which the current long-term bandwidth demands for each outgoing link will be superseded by the new estimates (again based on the information provided by the CORE Server along with the route update messages). This, permits the **CORE MAC Module** to react to impending changes in long-term bandwidth requirements for the outgoing links and issue appropriate bandwidth requests and initiate the handshakes needed to reserve bandwidth (see Sec. 8.5.2 for a detailed discussion).

8.6 Summary

In this chapter we have had a detailed look at the CORE Architecture, its various components, and how these interact with each other to implement the routing solution computed centrally at the CORE Server. We will next present a thorough evaluation of different aspects of CORE and look at its performance in WMNs in various operating scenarios.

9 Evaluation

We next evaluate CORE's heuristics via monte-carlo simulations (CORE's functionality was implemented into an extended version of the JiST/SWANs simulator [10]). Here, we first study the performance of CORE in a realistic mesh topology shown in Fig. 9.1 for three distinct simulation setups. Preliminary versions of the evaluation have been presented in [74, 73].

In Setup I we compare the quality of the solution obtained using CORE vs. the optimal solution (a brute-force approach considering all possible route combinations in the working set of the heuristic). In Setup II we analyze the performance of CORE for different usage scenarios of the WMN. In particular, we investigate different traffic distributions to model an access network, an enterprise/community network and a mixture of both. In Setup III we present an alternative flow sorting strategy for the *OptRC* algorithm which draws on the insights obtained from the results for Setup II. We investigate the influence of the new flow sorting strategy on the performance of CORE for the same settings as in Setup II. As a baseline for our joint routing and scheduling heuristic, we use standard shortest-path routing using either hop-count or the blocking-cost of a path as defined in Eq. (8.3) as the routing metric. Finally in Sec. 9.4 we look at some vital aspects of CORE in operation, in particular, we focus on how CORE is able to deploy the centrally computed solutions using default distributed components present in the WMN. Additional results are presented in Sec. C.2 to further study in depth individual issues in

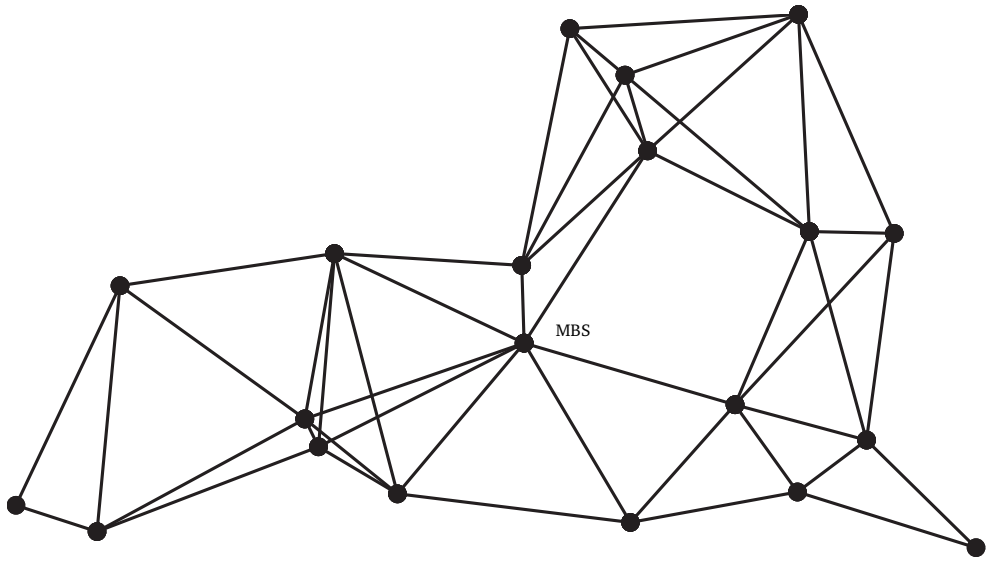


Figure 9.1: Simulated 20 Node WMN topology with Mesh Base Station (MBS)

the working of CORE. These results support the findings presented in this chapter.

9.1 Simulation Results: Setup I

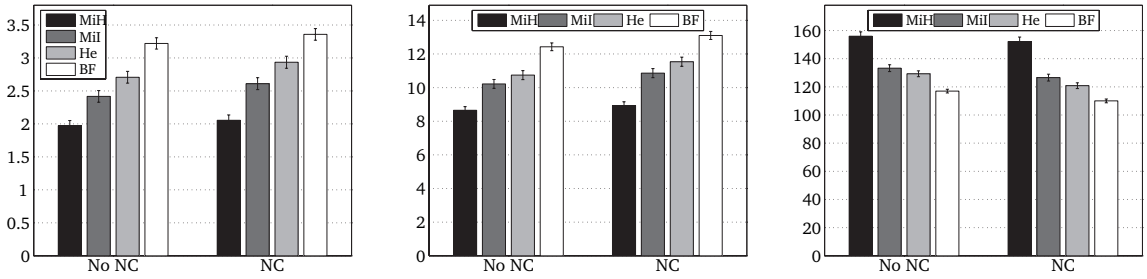
To find the globally optimal solution we implemented a brute-force algorithm which explores all feasible route combinations. Due to the prohibitive computational costs involved for the brute-force search we limited the study in Setup I to only 6 flows in the network with a limit of 20 minislots in the data-subframe. The traffic demands for the individual flows were uniformly dis-

tributed between 2–7 minislots per frame per flow. The source destination pairs were randomly selected such that trivial flows were not generated (i.e. the minimum hop-path is of at least two hops length). We performed 400 replications of the experiment. The following algorithms have been studied:

- Minimum-hop routing (`MiH`)
- Minimum-blocking path routing (`MiI`, see Eq. (8.3))
- CORE's routing heuristic (`He`)
- Optimal routing using brute-force (`BF`)

We analyzed setups with and without network coding (`NC`/`No NC`). In the `NC` case, network coding opportunities are recognized by the scheduler and the corresponding slots for the (multicast) transmission are reserved.

Fig. 9.2(a) shows the average *number of flows which could be scheduled* (we show the 95% confidence intervals if not noted otherwise). Using the optimal `BF` algorithm on an average less than 4 of the 6 flows offered could be scheduled, i.e. we operate the network in saturation. Fig. 9.2(a) shows that the shortest-path routing `MiH` performs worst. `MiI`, the second worst scheme, outperforms `MiH` because the selection of minimum-blocking paths frees network resources, thus enabling the scheduling of additional flows. This result acknowledges the importance of choosing interference-aware routes in the WMN. The developed heuristic `He` performs even better. Infact, the increase in performance compared to `MiI` indicates the performance gain that can be realized by optimizing the routes for all flows in parallel. Moreover, the performance of `He` is close to 90% of the optimal performance of `BF`, if we consider the number of admitted flows.



(a) Number of flows scheduled (b) Demand served (c) Blocked per demand

Figure 9.2: Simulation Results for Setup I

The obtained results with network coding are in line with our earlier findings for the relative performance differences of the analyzed schemes. In absolute terms, despite the fact that we only offer 6 flows, the creation of network coding opportunities helps reduce the number of transmissions, thereby leading to fewer blocked links per minislot, thus supporting more flows. We see that the performance of `He` is even closer to the optimum compared to the non-network coding case.

Fig. 9.2(b) shows the average *served total traffic demand* (in minislots required per frame) for the scheduled flows. It can be seen that `He` supports a larger traffic demand compared to `MiH` and `MiI`. Again, `BF` gives the optimum performance that can be achieved. Fig. 9.2(c) shows the number of (link, minislot) tuples *blocked per scheduled demand*, which describes the bandwidth efficiency of the scheme. A lower value for this metric indicates a higher efficiency of the scheme. This translates into a higher probability that available (link, minislot) tuples still exist in the network for a fixed demand; these available resource can in turn be used for scheduling additional data. Only `BF` is able to outperform `He`.

Table 9.1 shows the number of route *combinations tried* (searched) by the individual algorithms to achieve the results shown in Fig. 9.2. The value *reroutes* represents the number of times a flow was routed on the non-default path (which is assumed to be the `MiI` path). Given, the small number of flows in Setup I we do not have a large number of reroutes. It can be seen that the `He` and `BF` algorithms reroute more flows in order to establish network coding opportunities, thereby conserving bandwidth in the network. The value *NC sessions* counts the mean number of network coding sessions established (see Sec. 5.1 for the formal definition of a network coding session). Even for the small setup, our algorithm `He` is able to establish significantly more NC sessions than the baseline algorithms.

Table 9.1: Search Effort, Successful Reroutes, and Established Network Coding Sessions for Setup I

	Combinations Tried		Reroutes		NC Sessions	
	No NC	NC	No NC	NC	No NC	NC
MiH	6 ± 0	6 ± 0	0	0	0	0.38 ± 0.08
MiI	6 ± 0	6 ± 0	0	0	0	0.95 ± 0.14
He	74 ± 1	72 ± 1	0.845 ± 0.08	0.95 ± 0.09	0	1.49 ± 0.15
BF	15624 ± 0	15624 ± 0	1.15 ± 0.10	1.24 ± 0.10	0	1.83 ± 0.15

9.2 Simulation Results: Setup II

In Setup II we analyze the performance of our heuristic for different traffic patterns, representing the following typical usage scenarios for WMNs:

- Operation of the WMN as a wireless access network is denoted as *AccNet*.
- Operation of the WMN with internal traffic only is denoted as *Intern*.
- Operation of the network in a hybrid/mixed setup is denoted as *Mixed*.

To model these scenarios, we introduce three different classes of traffic: *Internet traffic (Inet)*, *symmetric traffic (Sym)* and *asymmetric traffic (Asym)*. In each *Inet* connection the MBS is a communication endpoint; we assume lower bandwidth for the uplink than for the downlink. *Sym* flows always request the same amount of bandwidth for both directions of a source-destination pair, thus modeling, e.g. VoIP traffic. *Asym* traffic represents file transfers or video streaming sessions inside the network, with most of the demand in one direction and only a negligible amount in the reverse direction.

We instantiate the three modelled scenarios as shown in Table 9.2. The traffic pattern for the *AccNet* scenario consists of *Inet* flows only. In the *Intern* scenario we combine the *sym* and *asym*[*] traffic pattern for communication among nodes of the WMN. The *Mixed* scenario combines all three types of traffic. Table 9.2 shows the individual setting for the traffic patterns for all studied scenarios, whereby the demand is uniformly distributed and specified as demand in minislots per frame per flow. The demands for the different WMNs scenarios were chosen such that the total demand for all flows in each scenario was in average around 80 minislots per frame. We assumed a total number of 67 available minislots, which corresponds to about 70% of all minislots in the ETSI (n = 8/7, 3.5 MHz, OFDM 256) mode of the IEEE 802.16 standard.

[*] Asym. traffic is modelled with a reverse demand of zero, as the negligible demands are scheduled using short-term reservations not controlled by CORE, further for the reservation-based MAC a non-zero amount for the reverse demand has only low impact as soon as the reservation is issued.

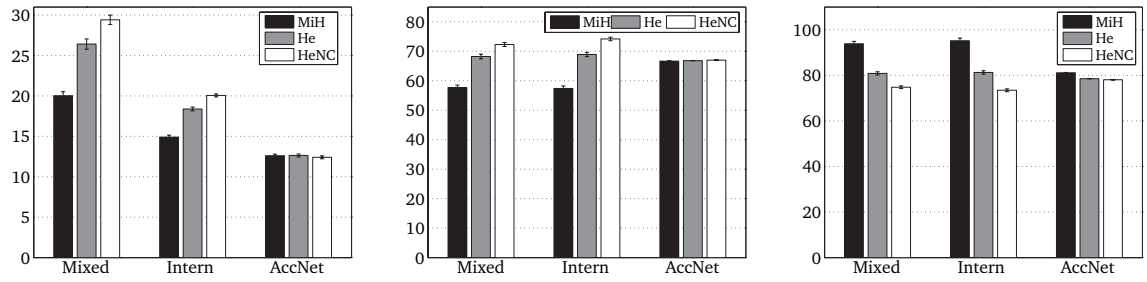
Table 9.2: Traffic Patterns for Setup II

		Inet Up./Down.	Sym	Asym
AccNet	No. of flows	11 / 11	0	0
	Demand per flow	1-2 / 4-8	0	0
	Mean total demand	82.5 minislots per frame		
Intern	No. of flows	0	2x 8	6
	Demand per flow	0	3-4	3-5
	Mean total demand	80 minislots per frame		
Mixed	No. of flows	11 / 11	2x 5	5
	Demand per flow	1 / 1-3	3	2-5
	Mean total demand	80.5 minislots per frame		

Table 9.3: Search Effort, Successful Reroutes and Established Network Coding Sessions for Setup II

	Combinations Tried			Reroutes			NC Sessions		
	Mixed	Intern	AccNet	Mixed	Intern	AccNet	Mixed	Intern	AccNet
MiH	37 ± 0	22 ± 0	22 ± 0	0	0	0	0	0	0
He	912 ± 6	531 ± 6	595 ± 4	11.0 ± 0.4	6.8 ± 0.2	3.2 ± 0.1	0	0	0
HeNC	870 ± 8	500 ± 5	578 ± 3	12.5 ± 0.4	7.5 ± 0.3	3.2 ± 0.1	14.0 ± 0.3	13.3 ± 0.3	1.9 ± 0.2

Fig. 9.3 and Table 9.3 show the results for `MiH`, `He` (CORE's heuristics with no network coding used) and `HeNC` (CORE's heuristics with network coding (NC) enabled) for Setup II[†]. The results clearly indicate the superior performance of `He` vs. the baseline `MiH` if traffic patterns leave room for optimization.



(a) Number of flows scheduled (b) Demand served (c) Blocked per demand

Figure 9.3: Simulation Results for Setup II

In particular, for the *Intern* scenario, `He` is able to admit around 20% additional flows compared to `MiH`, while `HeNC` realizes an improvement of about 33% in scheduled flows. For the *Mixed* scenario, the improvements are even more impressive: `He` outperforms `MiH` by scheduling around 30% additional flows, `HeNC` is able to increase the number of flows by nearly 50%. At the same time, Fig. 9.3 shows that *AccNet* does not provide this room for optimization, which is due to the bottleneck MBS. Since we assumed a total of 67 minislots, the MBS can serve a maximum demand of 67 minislots per frame, which also limits the performance of all three schemes

[†] We omit the presentation of the results for the network coding variant `MiHNC`, because the inherent limitations in identifying NC sessions leads only to marginal performance gains relative to `MiH`.

as shown in Fig. 9.3 (b). Network coding does not help in this scenario either, because our heuristic prefers flows with high demand, i.e. downstream flows and does not permit sufficient upstream flows to the MBS to yield network coding opportunities. In contrast, as shown earlier, the possible gain further improves if network coding is enabled (`HeNC`) and sufficient network coding opportunities can be identified, which is true for the scenarios *Mixed* and *Intern*.

9.3 Simulation Results: Setup III and *Nearness*

The results in Sec. 9.2 show that for the AccNet scenario `HeNC` does not yield performance gains over the `HE` algorithm. Thus, the additional activation of network coding using CORE's `HeNC` heuristic was ineffective. This can be attributed to the default ordering of flows based on their demand, which was used for the *OptRC* algorithm that is part of both heuristics. Due to this ordering of flows, the heuristics always chose to jointly optimize the routes for the higher throughput flows together before considering the flows with lower throughput. In the AccNet scenario, the flow setup was chosen to reflect the typical scenario in access networks, i.e. asymmetric Internet traffic with the volume of download traffic (traffic from the MBS towards the SSs) being much higher than the volume of upload traffic. Sorting the set of flows in the network in descending order of their demands results in the download flows to appear earlier in the sorted list, while the small upload flows appear at the end. Based on this sorting, CORE's heuristic jointly optimizes the routes for the downlink flows in parallel, but as these are all flows originating at the MBS (ingress/egress point for external traffic into/out of the WMN), the packets belonging to these flows are usually routed along a tree rooted at the MBS to the individual SSs. Thus, almost no latitude w.r.t. obtaining network coding opportunities by jointly optimizing the routes for the downlink flows exists. After having scheduled the high throughput flows using the central server, the share of the centrally managed bandwidth is nearly void. As a result the MBS can only process very few uplink flows of lower demand. The flows not fitting into the envelope assigned for centralized management are then left to be scheduled using the rest of the bandwidth in a distributed manner. Hence, for the studied AccNet scenario, `HeNC` does hardly setup any network coding sessions.

Obviously, if the system schedules and optimizes a pair of an uplink and its corresponding downlink flow such that the flows are routed along the same nodes (however in opposite directions) then a number of network coding opportunities can be created at the relay nodes (similar to that shown in Fig. 2.2). To identify and jointly combine such pairs of flows (not necessarily between the same source and destination nodes) we need to depart from the demand-based flow ordering.

We propose a novel metric for determining the order of the flow, which we term as *nearness*. The intuitive idea behind nearness is to identify flows which are topologically close to one another in the WMN. Such flows are suitable candidate flows for optimizing their routes in parallel. To be more precise, the source node of one flow must be close (in the WMN topology) to the destination node of the second flow and vice versa. The most trivial case for this can be seen for the AccNet flows, where the source node of the uplink flow is the destination node for the downlink flow and the source node for the downlink flow is destination node for the uplink flow.

We define the *nearness* of two flows as in Eq. (9.1).

$$\delta\left(f_x, f_y\right) = \frac{h\left(s_x, d_y\right) + h\left(s_y, d_x\right)}{2} \tag{9.1}$$

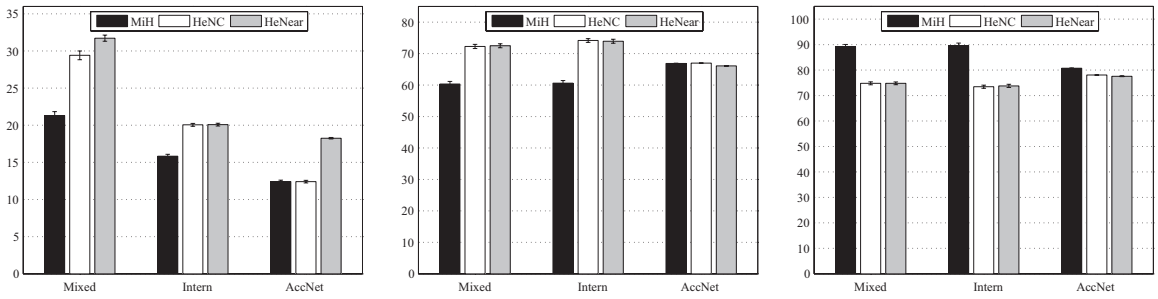
Where f_x and f_y are two flows with sources s_x and s_y and destinations d_x and d_y, respectively. $h(s,d)$ is the length of the minimum hop path from s to d. We can modify the OptRC Algorithm (Algorithm 1) to use the *nearness* metric when adding flows to a partition, by replacing lines 6–10 of Algorithm 1 by the following algorithm:

Algorithm 4 Flow Selection Using *nearness* Heuristic

1: $x \leftarrow 0$
2: **while** $comb \cdot k_x \leq C$ **do**
3: $\quad comb \leftarrow comb \cdot k_x$
4: $\quad p \leftarrow p \cup \{f_x\}$
5: $\quad f \leftarrow f \setminus \{f_x\}$
6: $\quad x \leftarrow i$ such that $\delta\,(p_0, f_i)$ is minimal $\forall f_i \in f$
7: **end while**

As discussed in Sec. 8.4.2, p_0 is the most important flow in a partition. Instead of the flow with the highest demand when using the default metric, the *nearness* metric considers the flow that is nearest to all the other flows in its partition as p_0.

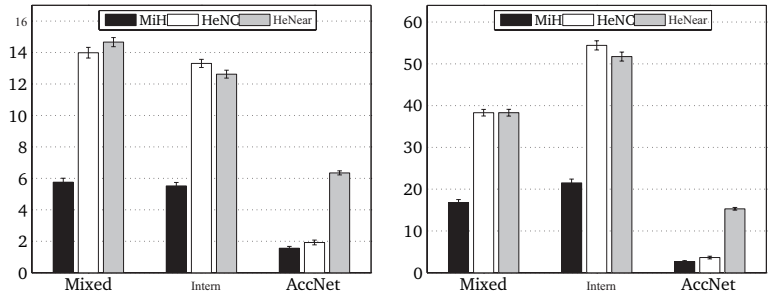
For the results presented in this section we compare the performance of the following algorithms `MiH`, `HeNC`, and `HeNear`. Where `HeNear` is CORE's heuristic with network coding enabled, and employing the *nearness* metric together with the correspondingly modified *OptRC* algorithm. Here, the main aim is to investigate the influence of the usage of the nearness metric on the capability of CORE to find and utilize network coding opportunities. For the results presented in Setup III, we use traffic settings similar to Setup II. We keep all other simulation parameters constant, except for the use of the nearness metric for `HeNear`. As both the heuristics are bound by the same computational costs, and there is no change to the computational costs incurred by the `MiH` algorithm, we refrain from presenting the computational costs for the simulations in Setup III.



(a) Number of flows scheduled (b) Demand served (c) Blocked per demand

Figure 9.4: Simulation results for Setup III

Fig. 9.4 shows the results for `MiH`, `HeNc`, and `HeNear` for Setup III. As can be observed, the performance `HeNear` equals or betters the performance of `HeNC` in all the operating scenarios. Notably, significantly more flows are admitted into the network by `HeNear` in the *Mixed* and the *AccNet* scenarios. However, this does not translate to more total demand served. For the investigated setting we observe that `HeNear` schedules more lower demand flows which are *near* to some higher demand flows in the current partition of flows as compared to `HeNC` which prefers to first optimize the routes for all the higher demand flows in parallel before considering lower demand flows.



(a) Number of NC sessions established

(b) Number of slots reserved for Multicast NC sessions

Figure 9.5: Simulation results for Setup III

Fig. 9.5 shows the comparative performance of `MiH`, `HeNC`, and `HeNear` considering their ability to setup network coding sessions in the WMN. As expected, `HeNear` is able to setup significantly more network coding sessions[‡] for the *Mixed* and *AccNet* scenarios. The gain in the number of network coding sessions setup is mainly due to the joint optimization of routes for traffic flows which form uplink/downlink pairs, but also other topologically close flows. However, the *nearness* metric is not necessarily always effective in setting up network coding sessions that further improve the network capacity. This can be seen from the slightly worse performance of `HeNear` when compared to `HeNC`. We can explain this result, because even if the source of one flow is close to the destination of another flow and vice versa, this does not imply that the individual nodes on the already established routes are close to each other. Rerouting the flows slightly thus does not necessarily allow the setup of significantly higher number of network coding sessions.

We observe a similar trend in the overall number of slots reserved (per frame) for the network coding sessions setup in the WMN. Although the number of network coding sessions setup in the *Mixed* scenario is slightly higher for `HeNear` as compared to `HeNC`, the total number of slots which are reserved for the network coding sessions is very close for both schemes. Again, this is possible, as `HeNC` tends to optimize the routes for the higher throughput flows in parallel first. Thus, it may e.g. setup a single NC session where the slots reserved for coding are 10 slots per frame. On the other hand, `HeNear` tends to optimize routes for nearby flows first, and may thus e.g. set up 10 NC sessions for 10 uplink/downlink flow pairs. Since the traffic may be asymmetric in volume for the latter case, flows are matched for network coding that e.g. need 1 slot per frame of uplink bandwidth, and each 10 slots per frame of downlink bandwidth. Hence due to the asymmetry in the traffic, at most one slot per frame will be attributed to network coding for each of the established NC sessions.

We can conclude that using the *nearness* metric one can very efficiently identify and select uplink/downlink or similar flow pairs for jointly optimizing the routes. This leads to a significantly increased number of network coding opportunities using `HeNear`, especially in the *AccNet* scenario. In contrast, in the *AccNet* scenario in Setup II, `HeNC` was not able to show any significant performance gains over `He`. However, to achieve a significant capacity gain in the network, `HeNear` relies not only on the proposed flow sorting, but also needs to operate under network and traffic parameters such as topology restrictions and flow asymmetry enabling the necessary latitude for rerouting of traffic.

[‡] A network coding session consists of a node (relay) which transmits coded packets to its neighbours thereby relaying the packets via network coding instead of using simple store-and-forward

9.4 Simulation Results: CORE in Operation

In the previous sections, i.e. Sec. 9.1, Sec. 9.2, and Sec. 9.3 we evaluated the quality of the heuristics computed by the CORE Server in different settings. However, as discussed, CORE relies on distributed components, i.e. components at each node in the WMN for the final deployment of network coding, as well as for the reservation of bandwidth and management of the transmission schedules. In this section we will look at selected results which demonstrate the critical aspects of CORE's working, and show that CORE is in fact able to approach the centrally computed solutions via distributed components.

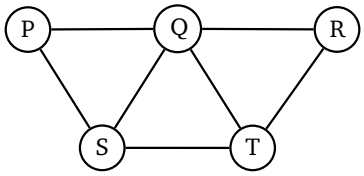


Figure 9.6: Toy Topology

We explained with a running example and toy topology shown in Fig. 9.6 the overall operation of CORE in a nutshell. Let us refer to Fig. 9.6, and assume the same network frame parameters for the IEEE 802.16 MeSH mode as specified in Sec. 9.2. Let us assume that we have the following flows in the network: Flow 1: from node P, to node R starting at time 10 sec and stopping at 20 sec with a constant data rate (demand) equivalent to 6 minislots per frame; and Flow 2: from node T to node P, starting at 13 sec and stopping at 17 sec with a demand equivalent to 11 minislots per frame.

Initially, the WMN will use the default routes provided by the routing in place in the network (i.e. in our case shortest hop paths). Thus, Flow 1 is routed via the path $P \rightarrow Q \rightarrow R$, which, incidentally is also the minimum interference path, and will provide the best delay when sufficient bandwidth is reserved all along the path. Thus, after the initial bandwidth requests have been processed (affected by the control delay and overhead which we discussed in depth in Chap. 3 and Chap. 4), the mean end-to-end delay for packets belonging to Flow 1 comes down to 20ms as shown in Fig. 9.7 at point (1).

This is in our setup the lowest delay which would be achievable for this path assuming that data usually cannot be forwarded in the same frame in which it has been received by a node from the previous hop. The frame duration for the selected IEEE 802.16 parameters is 10ms. Now, when Flow 2 starts at time $t = 13s$, Flow 2, similar to Flow 1 will be routed along the default shortest hop path (here, route $T \rightarrow S \rightarrow P$). The central CORE server, here node Q, is then informed about the demand of this flow too, and uses its heuristics to determine that the overall minimum interference route set uses route $P \rightarrow S \rightarrow T \rightarrow R$ for Flow 1 and $T \rightarrow S \rightarrow P$ for Flow 2.

This increases the path length for Flow 1 by one hop, however, the rerouting of the flows as per the routes computed by the heuristics at the central server reduce the overall interference in the network by allowing node S to act as a network coding relay for packets flowing in both directions via the node between the nodes P and T. As discussed in Sec. 7.2 the CORE server then sends appropriate control messages to the nodes in the WMN to instruct them to update their routing tables and use the routes proposed by the CORE server for the individual flows instead of the default routes. Thus, the CORE server basically provides an extended routing functionality adapting the default routes where needed, hence the name Centrally Optimized Routing Extensions.

However, in addition to notifying the individual nodes to adapt their routing tables, the CORE server also instructs the nodes to adapt their bandwidth reservations on the individual links in

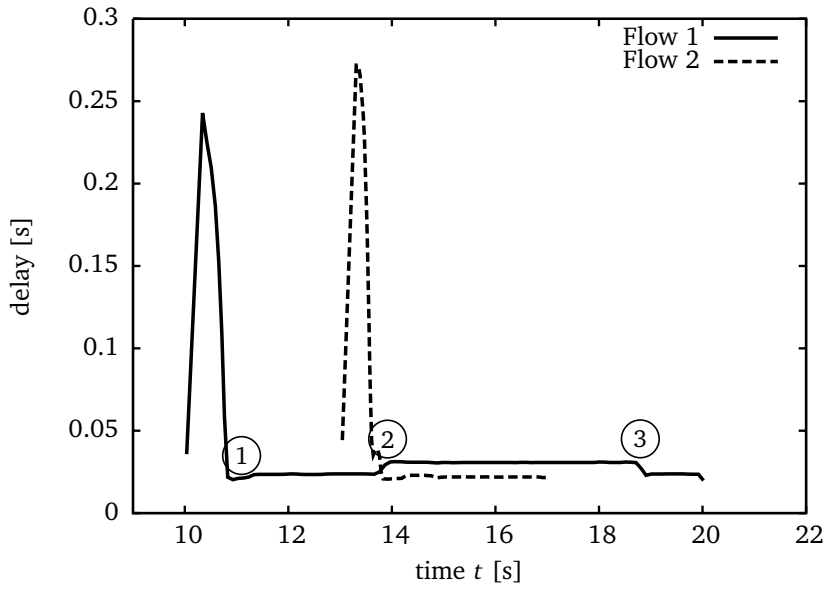


Figure 9.7: End-to-end Delays for the Toy Topology from Fig. 9.6

advance to the changes of the routing tables. It should be noted that besides these instructions which are provided via a cross-layer interface, CORE does not play any role in the actual bandwidth reservations nor does it reserve multicast bandwidth for network coding, or control network coding operations centrally, but the distributed bandwidth reservation mechanisms present on the node in the MeSH mode are used. We used SONC (see Part II of the thesis) as the default means for deploying network coding. Note, we assume that these default components only needed to be slightly modified to both enable interpretation of CORE's instructions, as well as to enable multicast bandwidth reservation and network coding functionality if these are missing. The interested reader is referred to Refs. [1, 115, 84, 116] for additional details about the bandwidth reservation as well as network coding solutions for reservation based WMNs.

Thus, based on the route updates received from the server, the nodes will reroute Flow 1 such that it now takes a longer route. This is reflected by the slight increase in the end-to-end delay of packets for this flow (see (2) in Fig. 9.7). However, due to the additional advance bandwidth reservation change instructions sent by the CORE server, the transition from one path to the other is relatively smooth, and does not incur the comparatively high initial delays due to the time required for bandwidth reservation along the path. As Flow 2 ceases at $t = 17$s, Flow 1 is rerouted back to its default path ((3) in Fig. 9.7). We used the toy-topology to be able to easily show the effect of the route change for flows, and how the CORE framework mechanisms allow a smooth transition from one path to another for the flows. For the time duration where both the flows are active in the network, and there exists a cross-flow at node S, this node may set up a network coding session allowing network coding to be deployed. The mechanisms for network coding itself are however beyond the influence of CORE, and there are different possible mechanisms to implement network coding. For the results presented here we used SONC as described in Part II of this thesis. In general, as we have shown in our work Ref. [80, 82] in reservation based systems, network coding is meaningful only for long-lived flow pairs where the additional reservation overhead can be amortized by coding a large number of packets arriving over time for the chosen flow pair/set. We highlighted this aspect in detail in Chap. 4.

We now once again turn back to the bigger topology shown in Fig. 9.1. The aim is to see if CORE in operation does indeed bring the expected increase in terms of additional network

coding sessions which are enabled in the WMN for a given set of flows in the WMN. We use the same IEEE 802.16 frame parameters as considered in Sec. 9.2, and look at the Mixed Traffic scenario specified in Tab. 9.2. Thus the flow parameters are the same as used for the Mixed Traffic Scenario there. We again assume that the CORE server's *OptRC* heuristic sorts flows based on their demands. For this setup, based on the computation at the CORE server, we see from Fig. 9.5 (a) that CORE using the heuristics (i.e. `HeNC`) is expected to enable on an average 14 network coding sessions, which when compared to the around 6 network coding sessions expected to be enabled by the standard routing in the WMN is a significant improvement.

We simulated the operation of the network for a total of 11000 frames, with the Mixed Traffic scenario such that the individual flows started at random times between frame 2000 and 4000 and then existed for the rest of the simulation. We used a IEEE 802.16 MAC layer with distributed bandwidth reservation mechanisms (see Ref. [84]) with additional extensions supporting SONC. As discussed earlier SONC will start a network coding session and actively code packets belonging to a network coding constellation (stream/flow pair) only when it observes that the flows have existed for a certain minimum duration, and have sufficient traffic demand to profitably gain from network coding (see discussion in Sec. 5.3.2).

Fig. 9.8 shows the number of network coding sessions established using the implemented network coding enabled MAC layer. Fig. 9.8 shows that the total number of network coding sessions established (i.e. for which the nodes acting as relays were able to observe cross-flows with sufficient traffic demand, and where these flow's existed long enough) in the WMN when using CORE is more than double the number of network coding sessions which could be established in the WMN for the same set of flows without CORE in operation. Furthermore one sees that with more and more flows entering the network (between frames 2000 and 4000) the number of possible network coding sessions increases for both the WMN with and without CORE. This is quite possible as some default routes may lead to suitable network coding opportunities. Additionally, one also sees that the network coding sessions are established with a certain lag to the flow's entering the network. This is due to the implementation we used where network coding sessions were established only once the flows had existed for a minimum time duration and were observed to have a minimum traffic demand. But, in both the cases, i.e. when operating with CORE enabled, and without CORE enabled, the MAC layer implementation was the same, so these effects are similar for both scenarios. One sees that the CORE server adapts the routes for the flows multiple times (i.e. with the entry of flows where it finds out that another route combination may lead to a schedule blocking less minislots, similar to the case seen for the toy-topology discussed earlier). This is seen by the steady step-wise increase in the number of network coding sessions which could be established.

Now, if one compares the number of network coding sessions which the WMN was able to establish with the number of network coding sessions which had been estimated by the computations at the CORE server for the scenario (see Fig. 9.5 (a)), we see that the distributed means of implementing CORE's centrally computed solutions come close to the central solution in terms of the network coding sessions which could be established. The difference to the central solution is only around two sessions in this simulation, and a SONC implementation which relaxes some of the constraints discussed in Sec. 5.3.2 would be less conservative in establishing network coding sessions would further reduce this difference. The conservative nature of our SONC implementation is especially seen when we look at the number of network coding sessions which are established without CORE, here too the number of network coding sessions is slightly less than that which is theoretically possible. However, we have seen that in reservation based WMNs network coding involves considerable overhead for coding as well as maintenance and reservation of suitable schedules for transmitting packets involved in network coding, and hence should be deployed only when the constraints in Sec. 5.3.2 are satisfied.

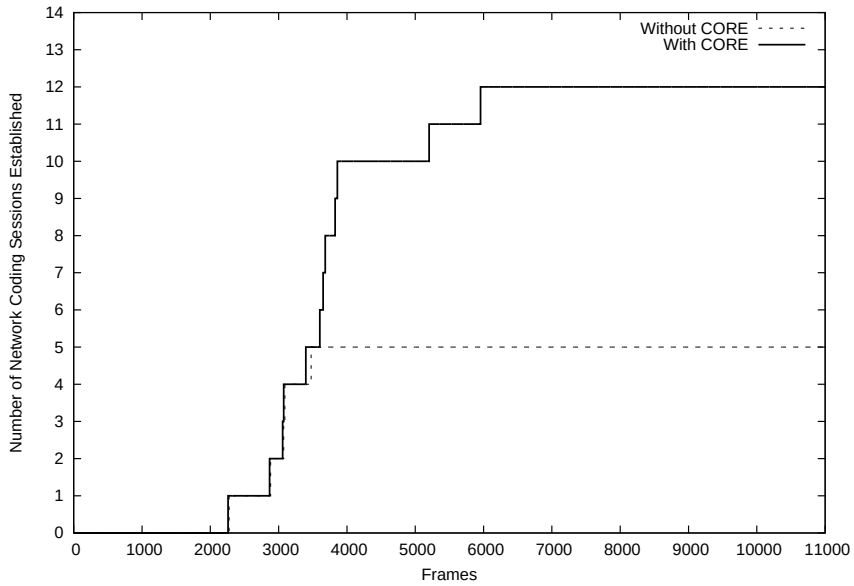


Figure 9.8: Network Coding Sessions Setup Over Time With and Without CORE

The two scenarios studied in this section show that the CORE mechanism works as expected within the simulator. Flows are rerouted in order to free network capacity whereby network coding opportunities are detected and used. Once a flow is admitted to the network by the BS the flow's end-to-end delay drops to values that correlate with the route length of the flow. Due to the Per_{∞} reservations used the delay stays nearly constant, even if other flows lead to a congestion of a node on the route.

Requesting bandwidth before a reroute actually takes place allows a smooth transition from one route to another. Further, we see that CORE is able to use the distributed components in the network successfully to approach the centrally computed solution.

9.5 Summary of CORE Results

The evaluation shows that CORE can very much improve the performance in WMNs. In networks that are tractable, our devised heuristics perform slightly worse than optimal solutions. However, please note that the former have been tuned such that they are able to operate in near real-time, while the latter are infeasible for realistic scenarios because of their runtime. Our scheme shows an excellent performance in realistic environments. In particular, the `He` as well as the `HeNC` variant significantly outperform the baseline of minimum-hop routing (`MiH`) by up to 50% improvement in terms of scheduled flows. At the same time, our algorithms are able to identify network coding opportunities and are practical to deploy network coding in TDMA/TDD mesh networks, even if these networks deploy per-hop link-level encryption.

The evaluation shows that CORE can very much improve the performance in WMNs. In networks that are tractable, our devised heuristics perform slightly worse than optimal solutions. However, please note that the former have been tuned such that they are able to operate in near real-time, while the latter are infeasible for realistic scenarios because of their runtime. Further, the network operator can trade off more computational complexity for better solutions, if the additional computational resources and time are available in the WMN. CORE shows an excellent performance in realistic environments. In particular, the `He` as well as the `HeNC` and the `HeNear` variants significantly outperform the baseline of minimum-hop routing (`MiH`) by up to 50% improvement in terms of scheduled flows. At the same time, our algorithms are

able to identify network coding opportunities and are practical to deploy network coding in TDMA/TDD mesh networks, even if these networks deploy per-hop link-level encryption.

Part IV

The Finale

10 Conclusions and Future Work

In this chapter we will briefly summarize the major contributions of the thesis and the vital insights which can be obtained from this work. Based on this we also shortly outline a set of interesting issues which are promising areas for further research.

10.1 Summary and Conclusions

This thesis has been concerned with the investigation of efficient means to support bandwidth savings in reservation based WMNs using network coding. Here, the thesis has shown analytically that the contemporary packet-by-packet approaches to network coding if applied to reservation based WMNs are highly insufficient and will perform poorly. Starting with this insight derived in the thesis, we introduced the new paradigm of Stream Oriented Network Coding (SONC) in this thesis. Using SONC nodes in the WMN make the network coding decisions, reservations for transmitting coded data as well as coding operations at the scale of streams and not on a packet-by-packet basis. To achieve this we introduced the concept of a stream or local stream which are the basic units for SONC. We defined and introduced the concept of network coding constellations which enable a node in the WMN to identify and decide if network coding is meaningful for a given set of streams at a given relay node in the WMN.

Here, we have derived a set of fundamental constraints which need to be satisfied by a network coding constellation in order for it to be considered meaningful. This aids designers of similar systems to discard sets of streams for which network coding over these streams is not beneficial. We proved the need for the above constraints and additionally used these to design heuristics which enable to choose a subset of the meaningful network coding constellations for deployment. Here we make a vital distinction between network coding constellations requiring overhearing and such which do not need overhearing. To enable nodes which want to serve as relays for a network coding session we proposed, the lightweight degree of freedom metric to allow the nodes to quantify the benefits which they would obtain using network coding for the respective session. Additionally we also provide means to quantify the total degree of freedom of a WMN, which in turn permits us to measure relatively the costs of scheduling individual unicast as well as multicast transmissions in the WMN.

We designed and presented a modular framework which allows the efficient operation of SONC from identification of network coding constellations to setting up of network coding sessions, operation of coding over streams, and finally tearing down the established network coding session. For each of the above phases we designed and presented efficient solutions and heuristics. The entire work described above was implemented and tested using the IEEE 802.16 MeSH mode as a prototype for reservation based WMNs. To enable this a standard conform simulator for the MeSH mode was implemented with a focus on the MAC Common Part Sublayer functionality. We evaluated the feasibility of the proposed mechanisms using a thorough simulation study.

Thus, in the first half of the thesis the major focus has been on the design of the novel paradigm (SONC) for network coding in reservation based WMNs. SONC relies only on local information and uses a cross-layer approach to enable easy sharing of information across layers in the protocol stack. The solutions designed are suited for distributed deployment at all the individual nodes in the WMN. We however, also note that when SONC is operating in the WMN

without WMN wide information some promising opportunities to benefit from network coding cannot be realized. This is the focus of the latter half of the thesis.

Here we propose our framework CORE (Centrally Optimized Routing Extensions) which uses centralized mechanisms to enable the WMN to efficiently benefit from SONC. Here, we use a central server which uses heuristics designed to be efficient to compute a beneficial combination of routes for the individual long-term flows in the WMN. To deploy the computed routing changes in the WMN, CORE relies on distributed components available at the individual nodes in the WMN and also closely interacts with the SONC components present at each node in the network. However, CORE is not dependent on the presence of SONC in the network, neither does SONC rely on CORE for its working. This modular design makes our proposed system easily extendable with each component being replaced by another one still permitting the entire system to work. Further, the distributed deployment of the centrally computed solutions for the routes for the flows permits low overhead as no complete network state needs to be maintained at the central server. Additionally, the individual nodes (the distributed components) are responsible for effecting the routing changes as instructed by CORE, as well as for the scheduling of data and SONC. Thus when the central server running the CORE heuristics fails, the network is not affected as a whole and can carry on with the operations, although not as efficiently as when operating with CORE running. Our findings show that using CORE in addition to SONC leads to significant gains in the WMN in terms of the amount of traffic which the WMN can accommodate. We present the details of the heuristics of CORE and study different variants of the heuristics. The CORE Framework similar to the SONC modules was tested using the IEEE 802.16 MeSH mode as a prototype to permit the demonstration of a proof-of-concept. We tested the performance and benefits which are obtained via CORE using a thorough simulation study which validates our design. Thus, in the latter half of the thesis we have focussed on global (WMN wide) optimization of the benefits which can be obtained by deployment of SONC in reservation based WMNs.

Thus, the thesis has presented a unique combination of both distributed (acting with just local information) as well as centralized mechanisms for efficiently managing the bandwidth in reservation based WMNs and achieve gains in the traffic supported via use of network coding. This thesis is thereby one of the first if not the first to study in depth practical means to efficiently deploy network coding in multihop reservation based WMNs. We have also implemented our proposed solutions within the framework of an IEEE standard (the IEEE 802.16-2004 MeSH Mode) thereby proving that our solutions are implementable in realistic WMNs. This thesis has lead to a detailed investigation of the basic underlying reservation mechanisms in such WMNS which can be found in some of our prior work listed in Appendix F. Thus, we believe that the work in this thesis lays down the foundations for carrying out further interesting research in such WMNs. In the next section we shortly outline a selection of research opportunities for further work based on the work in this thesis.

10.2 Outlook

The work in this thesis brings up some interesting issues for further detailed research. For example, the centralized solutions presented by CORE could be replaced by completely distributed solutions. However, achieving this with the same efficiency as well as with very low additional overhead remains a challenging question.

In this thesis a primary consideration has been to not introduce additional undue delay due to network coding. This is desirable for all kinds of traffic in general, however, given that some traffic is more delay tolerant than others could throw up interesting extensions to SONC for reservation based WMNs. For example, such traffic may be backlogged at intermediate nodes to

create a dynamic buffer of packets which can be used for coding when at any given moment no packets from the real-time streams are available for coding. One may thus opportunistically use SONC also for packets which do not require any time sensitive handling, thereby enabling the benefits of SONC to be applicable to a wider spectrum of traffic in the WMN.

In a similar manner it would be interesting to extend SONC to partially reservation based systems, e.g. systems which provide a mix of reservations for guaranteed QoS traffic and contention based access to the medium at other times for non-critical traffic.

Another interesting issue for investigation would be to study the interaction and effect of various bandwidth reservation policies at the MAC layer on the performance of SONC. Finally, though we have studied SONC and CORE in a detailed and standard conform simulator it would be interesting to deploy SONC in real systems and test the robustness of the developed solutions in the real world.

Bibliography

[1] N. d'Heureuse, "Cross-Layer Bandwidth Optimization Scheme for IEEE 802.16 Wireless Mesh Networks," Master's thesis, TU Darmstadt, 2007.

[2] R. Ahlswede, N. Cai, S. R. Li, and R. W. Yeung, "Network Information Flow," *IEEE Transactions on Information Theory*, vol. 46, no. 4, pp. 1204–1216, July 2000.

[3] I. F. Akyildiz and X. Wang, "Cross-Layer Design in Wireless Mesh Networks," *IEEE Transactions on Vehicular Technology*, vol. 57(2), pp. 1061–1076, 2008.

[4] I. F. Akyildiz, X. Wang, and W. Wang, "Wireless Mesh Networks: A Survey," *Computer Networks*, vol. 47, no. 4, pp. 445–487, March 2005.

[5] M. Allman, V. Paxson, and W. Stevens, "TCP Congestion Control," RFC 2581, 4 1999.

[6] H. Balakrishnan, V. N. Padmanabhan, S. Seshan, and R. H. Katz, "A Comparison of Mechanisms for Improving TCP Performance Over Wireless Links," *IEEE/ACM Transactions on Networking*, vol. 5, no. 6, pp. 756–769, 1997.

[7] H. Balakrishnan, V. N. Padmanabhan, S. Seshan, and R. H. Katz, "A Comparison of Mechanisms for Improving TCP Performance over Wireless Links," in *ACM SIGCOMM 1996*, 1996.

[8] H. Balakrishnan, S. Seshan, E. Amir, and R. H. Katz, "Improving TCP/IP Performance over Wireless Networks," in *ACM Mobicom 1995*, 1995.

[9] A. Banchs and X. Perez, "Providing Throughput Guarantees in IEEE 802.11 Wireless LAN," in *IEEE WCNC 2002*, 2002.

[10] R. Barr, "JiST/SWANS," http://jist.ece.cornell.edu/docs.html, 2004.

[11] E. Belding-Royer and C. Toh, "A Review of Current Routing Protocols for Ad hoc Mobile Wireless Networks," *IEEE Personal Communications*, vol. 6, no. 2, pp. 46–55, April 1999.

[12] R. Braden, T. Faber, and M. Handley, "From Protocol Stack to Protocol Heap – Role-Based Architecture," in *Hot Topics in Networking*, 2002.

[13] P. J. Brockwell and R. A. Davis, *Introduction to Time Series and Forecasting*, Springer, Ed. Springer, 2002.

[14] M. Cao, W. Ma, Q. Zhang, X. Wang, and W. Zhu, "Modelling and performance analysis of the distributed scheduler in IEEE 802.16 mesh mode," in *MobiHoc '05*, 2005, pp. 78–89.

[15] S. Chachulski, M. Jennings, S. Katti, and D. Katabi, "Trading Structure for Randomness in Wireless Opportunistic Routing," in *SIGCOMM 2007*, 2007.

[16] K.-W. Chin, J. Judge, A. Williams, and R. Kermode, "Implementation Experience with MANET Routing Protocols," *ACM SIGCOMM Computer Communications Review*, vol. 32, no. 5, pp. 49–59, 2002.

[17] I. Cidon and S. Moshe, "Distributed Assignment Algorithms for Multihop Packet Radio Networks," *IEEE Transactions on Computers*, vol. 38, no. 10, pp. 1353–1361, October 1989.

[18] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, MIT Press and McGraw-Hill, Ed. MIT Press and McGraw-Hill, 2001.

[19] T. Cui, L. Chen, and T. Ho, "Energy Efficient Opportunistic Network Coding for Wireless Networks," in *IEEE INFOCOM 2008*, 2008.

[20] V. Dadia, "Bandwidth Reservation Strategies for Distributed Scheduling in the IEEE 802.16 Mesh Mode," Master's thesis, Indian Institute of Technology Kharagpur, TU Darmstadt, 2008.

[21] D. S. J. De Couto, D. Aguayo, J. Bicket, and R. Morris, "A High-Throughput Path Metric for Multi-Hop Wireless Routing," in *ACM/IEEE Mobicom 2003*, 2003.

[22] S. Deb, M. Effros, T. Ho, D. R. Karger, R. Koetter, D. S. Lun, M. Medard, and N. Ratnakar, "Network Coding for Wireless Applications: A Brief Tutorial," in *IWWAN 2005*, 2005.

[23] G. Di Caro and M. Dorigo, "AntNet: Distributed Stigmergetic Control for Communications Networks," *Journal of Artificial Intelligence Research*, vol. 9, pp. 317–365, 1998.

[24] G. Di Caro and M. Dorigo, "Antnet: a Mobile Agents Approach to Adaptive Routing," Universite Libre de Bruxelles, IRIDIA, Tech. Rep. 12, 1997.

[25] R. Draves, J. Padhye, and B. Zill, "Comparison of Routing Metrics for Static Multi-Hop Wireless Networks," *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 4, pp. 133–144, October 2004.

[26] R. Dube, C. D. Rais, K.-Y. Wang, and S. K. Tripathi, "Signal Stability-Based Adaptive Routing (SSA) for Ad hoc Mobile Networks," *IEEE Personal Communications*, vol. 4, pp. 36–45, 1997.

[27] C. Eklund, R. Marks, K. Stanwood, and S. Wang, "IEEE standard 802.16: a technical overview of the WirelessMAN air interface for broadband wireless access," *IEEE Communications Magazine*, vol. 40, no. 6, pp. 98–107, June 2002.

[28] C. Fragouli, J. Widmer, and J.-Y. Le Boudec, "Efficient Broadcasting Using Network Coding," *IEEE/ACM Transactions on Networking*, vol. 16(2), pp. 450–463, 2008.

[29] GNU, "GLPK (GNU Linear Programming Kit)," http://www.gnu.org/software/glpk/glpk.html, 2008.

[30] T. Goff, N. B. Abu-Ghazaleh, D. S. Phatak, and R. Kahvecioglu, "Preemptive Routing in Ad hoc Networks." in *ACM/ IEEE Mobicom 2001* , 2001.

[31] K. Graffi, P. S. Mogre, M. Hollick, and R. Steinmetz, "Detection of Colluding Misbehaving Nodes in Mobile Ad Hoc and Mesh Networks," in *IEEE Globecom 2007*, 2007.

[32] P. Gupta and P. R. Kumar, "The Capacity of Wireless Networks," *IEEE Transactions on Information Theory*, vol. 46, pp. 388–404, 2000.

[33] E. Hadjiconstantinou and N. Christofides, "An Efficient Implementation of an Algorithm for Finding K Shortest Simple Paths," *Networks*, vol. 34(2), pp. 88–101, 1999.

[34] HART Communication Foundation, "HART 7 Specification," Standard, September 2007.

[35] O. Heckmann, "A System-oriented Approach to Efficiency and Quality of Service for Internet Service Providers," Ph.D. dissertation, TU Darmstadt, 2004.

[36] G. Hiertz, S. Max, T. Junge, D. Denteneert, and L. Berlemann, "IEEE 802.11s - Mesh Deterministic Access," in *14th European Wireless Conference, EW 2008*, 2008, pp. 1–8.

[37] T. Ho and R. Koetter, "Online Incremental Network Coding for Multiple Unicasts," in *DIMACS Working Group on Network Coding 2005*, 2005.

[38] T. Ho, R. Koetter, M. Medard, D. Karger, and M. Effros, "The Benefits of Coding over Routing in a Randomized Setting," in *ISIT 2003*, 2003.

[39] G. Holland, N. Vaidya, and P. Bahl, "A Rate-Adaptive MAC Protocol for Multi-Hop Wireless Networks," in *ACM/IEEE Mobicom 2001*, 2001.

[40] M. Hollick, "Dependable Routing for Cellular and Ad hoc Networks," Ph.D. dissertation, TU Darmstadt, 2004.

[41] IEEE Computer Society and IEEE Microwave Theory and Techniques Society, "802.11 IEEE Standard for Local and metropolitan area networks, Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications," IEEE Std. 802.11, January 1999.

[42] IEEE, "802.16 Standard for Local and Metropolitan Area Networks, Part 16: Air Interface for Fixed Broadband Wireless Access Systems," IEEE Std. 802.16-2004, October 2004.

[43] R. Jain, *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*, Wiley, Ed. Wiley-Interscience, 1991.

[44] M. Johansson and L. Xiao, "Cross-Layer Optimization of Wireless Networks Using Nonlinear Column Generation," *IEEE Transactions on Wireless Communications*, vol. 5, no. 2, pp. 435–445, February 2006.

[45] D. Johnson, D. Maltz, and J. Broch, *DSR The Dynamic Source Routing Protocol for Multihop Wireless Ad Hoc Networks*. Addison-Wesley, 2001, ch. 5, pp. 139–172.

[46] V. Kanodia, A. Sabharwal, and E. Knightly, "MOAR: A Multi-Channel Opportunistic Auto-Rate Media Access Protocol for Ad Hoc Networks," in *First Annual International Conference on Broadband Networks*, 2004.

[47] S. Karande, Z. Wang, H. R. Sadjadpour, and J. J. Garcia-Luna-Aceves, "Network Coding Does Not Change the Multicast Throughput Order of Wireless Ad hoc Networks," in *IEEE ICC*, 2009.

[48] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Medard, and J. Crowcroft, "XORs in the Air: Practical Wireless Network Coding," *IEEE/ACM Transactions on Networking*, vol. 16, no. 3, pp. 497–510, 2008.

[49] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Medard, and J. Crowcroft, "XORs in the Air: Practical Wireless Network Coding," *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 4, pp. 243–254, October 2006.

[50] V. Kawadia and P. R. Kumar, "A Cautionary Perspective On Cross-Layer Design," *IEEE Wireless Communications*, vol. 12 (1), pp. 3–11, 2005.

[51] A. Keshavarz-Haddadt and R. Riedi, "Bounds on the Benefit of Network Coding: Throughput and Energy Savings in Wireless Networks," in *IEEE INFOCOM 2008*, 2008.

[52] A. Khreishah, C.-C. Wang, and N. B. Schroff, "Cross-Layer Optimization for Wireless Multihop Networks with Pairwise Intersession Network Coding," *IEEE Journal on Selected Areas in Communications*, vol. 27, pp. 606–621, 2009.

[53] M. Kodialam and T. Nandagopal, "Characterizing Achievable Rates in Multi-hop Wireless Networks: The Joint Routing and Scheduling Problem," in *ACM MobiCom 2003*, September 2003, pp. 42–54.

[54] R. Koetter and M. Medard, "An Algebraic Approach to Network Coding," *IEEE/ACM Transactions on Networking*, vol. 11, no. 5, pp. 782–795, 2003.

[55] A. Kumar, D. Manjunath, and J. Kuri, *Wireless Networks*, Elsevier, Ed. Morgan Kaufmann, 2008.

[56] J. F. Kurose and K. W. Ross, *Computer Networking A Top Down Approach*, Addison-Wesley, Ed. Pearson Addison-Wesley, 2009.

[57] C. R. L. and A. V. Santhanam, " Optimal Routing, Link Scheduling and Power Control in Multihop Wireless Networks," in *IEEE INFOCOM 2003*, 2003.

[58] S. R. Li, R. W. Yeung, and N. Cai, "Linear Network Coding," *IEEE Transactions on Information Theory*, vol. 49, no. 2, pp. 371–381, February 2003.

[59] Z. Li and B. Li, "Network Coding: The Case of Multiple Unicast Sessions," in *Allerton Conference on Communications 2004*, 2004.

[60] Z. Li and B. Li, "Network Coding in Undirected Networks," in *CISS 2004*, 2004.

[61] X. Lin and S. Rasool, "Distributed and Provably-Efficient Algorithms for Joint Channel-Assignment, Scheduling and Routing in Multi-Channel Ad hoc Wireless Networks," *IEEE/ACM Transactions on Networking*, vol. 17, no. 6, pp. 1874–1887, 2009.

[62] X. Lin and S. Rasool, "A Distributed Joint Channel-Assignment, Scheduling and Routing Algorithm for Multi-Channel Ad Hoc Wireless Networks," in *IEEE INFOCOM 2007*, May 2007, pp. 1118–1126.

[63] X. Lin and N. B. Shroff, "The Impact of Imperfect Scheduling on Cross-Layer Rate Control in Multihop Wireless Networks," in *IEEE INFOCOM 2005*, 2005.

[64] Y.-D. Lin and Y.-C. Hsu, "Multihop Cellular: A New Architecture for Wireless Communications," in *IEEE INFOCOM 2000*, 2000.

[65] J. Liu, D. Goeckel, and D. Towsley, "Bounds on the Throughput Gain of Network Coding in Unicast and Multicast Wireless Networks," *IEEE Journal on Selected Areas in Communications*, vol. 27, no. 5, pp. 582–592, 2009.

[66] J. Liu, D. Goeckel, and D. Towsley, "Bounds on the Gain of Network Coding and Broadcasting in Wireless Networks," in *IEEE INFOCOM 2007*, 2007.

[67] J. Liu, D. Goeckel, and D. Towsley, "The Throughput Order of Ad hoc Networks Employing Network Coding and Broadcasting," in *MILCOM 2006*, 2006.

[68] D. S. Lun, N. Ratnakar, R. Koetter, M. Medard, E. Ahmed, and H. Lee, "Achieving Minimum-Cost Multicast: A Decentralized Approach Based on Network Coding," in *IEEE INFOCOM 2005*, 2005.

[69] H. Lundgren, E. Nordström, and C. Tschudin, "Coping with Communication Gray Zones in IEEE 802.11b Based Ad hoc Networks," in *ACM WoWMoM 2002*, 2002.

[70] M. K. Marina and S. R. Das, "On-demand Multipath Distance Vector Routing in Ad Hoc Networks," in *IEEE ICNP 2001*, 2001.

[71] MIT Computer Science and Artificial Intelligence Laboratory, "Roofnet," http://pdos.csail.mit.edu/roofnet/doku.php, 5 2010.

[72] P. S. Mogre and et al., "Centrally Optimized Routing Extension," Patent Registration, 2008.

[73] P. S. Mogre, N. d'Heureuse, M. Hollick, and R. Steinmetz, "CORE: Centrally Optimized Routing Extensions for Efficient Bandwidth Management and Network Coding in the IEEE 802.16 MeSH Mode," *Wireless Communications and Mobile Computing, Special Issue: Architectures and Protocols for Wireless Mesh, Ad Hoc, and Sensor Networks*, Accepted for Publication.

[74] P. S. Mogre, N. d'Heureuse, M. Hollick, and R. Steinmetz, "CORE: Centrally Optimized Routing Extensions for the IEEE 802.16 MeSH Mode," in *IEEE LCN 2008*, 2008.

[75] P. S. Mogre, N. d'Heureuse, M. Hollick, and R. Steinmetz, "A Case for Joint Near-optimal Scheduling and Routing in TDMA-basedWireless Mesh Networks: A Cross-layer Approach with Network Coding Support," in *IEEE MASS 2007*, October 2007.

[76] P. S. Mogre, K. Graffi, M. Hollick, and R. Steinmetz, "A Security Framework for Wireless Mesh Networks," *Wireless Communications and Mobile Computing, Special Issue: Architectures and Protocols for Wireless Mesh, Ad Hoc, and Sensor Networks*, Accepted for Publication.

[77] P. S. Mogre, K. Graffi, M. Hollick, and R. Steinmetz, "AntSec: Securing Organically Growing Wireless Mesh Networks," Multimedia Communications Lab (KOM), TU-Darmstadt, Germany, Tech. Rep., March 2007.

[78] P. S. Mogre, K. Graffi, M. Hollick, and R. Steinmetz, "AntSec, WatchAnt and AntRep: Innovative Security Mechanisms for Wireless Mesh Networks," in *IEEE LCN 2007*, 2007.

[79] P. S. Mogre, M. Hollick, S. Dimitrov, and R. Steinmetz, "Incorporating Spatial Reuse into Algorithms for Bandwidth Management and Scheduling in IEEE 802.16j Relay Networks," in *IEEE LCN 2009*, 2009.

[80] P. S. Mogre, M. Hollick, M. Kropff, R. Steinmetz, and C. Schwingenschloegl, "A Note on Practical Deployment Issues for Network Coding in the IEEE 802.16 MeSH Mode," in *IEEE WiNC 2008, IEEE SECON 2008*, 2008.

[81] P. S. Mogre, M. Hollick, C. Schwingenschloegl, and R. Steinmetz, *WiMAX/MobiFi: Advanced Research and Technology*. Auerbach Publications, 2007, ch. QoS Architecture for Efficient Bandwidth Management in the IEEE 802.16 MeSH Mode, pp. 197–216.

[82] P. S. Mogre, M. Hollick, C. Schwingenschloegl, A. Ziller, and R. Steinmetz, *WiMAX Evolution*. Wiley-Interscience, 2009, ch. WiMAX Mesh Architectures and Network Coding, pp. 145–162.

[83] P. S. Mogre, M. Hollick, and R. Steinmetz, "The IEEE 802.16-2004 MeSH Mode Explained, KOM-TR-2006-08," KOM, TU Darmstadt, Germany, ftp.kom.tu-darmstadt.de/pub/TR/KOM-TR-2006-08.pdf, Tech. Rep., 2006.

[84] P. S. Mogre, M. Hollick, R. Steinmetz, V. Dadia, and S. Sengupta, "Distributed Bandwidth Reservation Strategies to Support Efficient Bandwidth Utilization and QoS on a Per-Link Basis in IEEE 802.16 Mesh Networks," in *IEEE LCN 2009*, 2009.

[85] P. S. Mogre, G. Vernet, M. Hollick, and R. Steinmetz, "The Design of a Forecast Based Bandwidth Reservation Request Manager for the IEEE 802.16 MeSH Mode," KOM, TU Darmstadt, Germany, ftp.kom.tu-darmstadt.de/pub/TR/KOM-TR-2007-09.pdf, Tech. Rep., 2007.

[86] S. Murthy and J. J. Garcia-Luna-Aceves, "An Efficient Routing Protocol for Wireless Networks," *Mobile Networks and Applications*, vol. 1, pp. 183–197, 1996.

[87] S. G. Nash and A. Sofer, *Linear and Nonlinear Programming*, McGraw-Hill, Ed. McGraw-Hill, 1996.

[88] M. J. Neely, E. Modiano, and C. E. Rohrs, "Dynamic Power Allocation and Routing for Time Varying Wireless Networks," in *IEEE INFOCOM 2003*, 2003.

[89] R. Nelson and L. Kleinrock, "Spatial TDMA: A Collision-Free Multihop Channel Access Protocol," *IEEE Transactions on Communications*, vol. 33, no. 9, pp. 934–944, September 1985.

[90] Network Working Group: J. Rajahalme, A. Conta, B. Carpenter and S. Deering, "IPv6 Flow Label Specification," Request for Comments: 3687, Standards Track, March 2004.

[91] P. Patras, A. Banchs, and P. Serrano, "A Control Theoretic Framework for Performance Optimization of IEEE 802.11 Networks," in *IEEE INFOCOM Workshops 2009*, 2009.

[92] C. E. Perkins and E. Belding-Royer, "Ad hoc On-Demand Distance Vector Routing," in *IEEE Workshop on Mobile Computing Systems and Applications*, 1999.

[93] C. E. Perkins, E. Belding-Royer, and S. R. Das, "Ad hoc On-Demand Distance Vector (AODV) Routing," IETF RFC 3561, July 2003.

[94] C. E. Perkins and P. Bhagwat, "Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers," in *ACM SIGCOMM*, 1994.

[95] J. Postel, "Transmission Control Protocol," RFC 793, 9 1981.

[96] V. T. Raisinghani and S. Iyer, "Cross-layer Design Optimizations in Wireless Protocol Stacks," *Computer Communications*, vol. 27, pp. 720–724, 2004.

[97] A. Raniwala, K. Gopalan, and T. Chiueh, "Centralized Channel Assignment and Routing Algorithms for Multi-channel Wireless Mesh Networks," in *ACM SIGMOBILE Mobile Computing and Communications Review (MC2R)*, 2004.

[98] B. Sadeghi, V. Kanodia, A. Sabharwal, and E. Knightly, "OAR: An Opportunistic Auto-Rate Media Access Protocol for Ad Hoc Networks ," *Wireless Networks*, vol. 11, no. 1, pp. 39–53, 2005.

[99] B. Sadeghi, V. Kanodia, A. Sabharwal, and E. Knightly, "OAR: An Opportunistic Auto-Rate Media Access Protocol for Ad hoc Networks," in *ACM Mobicom 2002*, 2002.

[100] Y. E. Sagduyu and A. Ephremides, "Joint Scheduling and Wireless Network Coding," in *First Workshop on Network Coding, Theory, and Applications (NetCod 2005)*, 2005.

[101] B. Scheuermann, W. Hu, and J. Crowcroft, "Near-Optimal Co-ordinated Coding in Wireless Multihop Networks," in *ACM Sigcomm CoNEXT 2007*, 2007.

[102] J. B. Schmitt, "Heterogenous Network QoS Systems," Ph.D. dissertation, TU Darmstadt, 2000.

[103] Seattle Wireless Project, "Seattle Wireless Network," http://www.seattlewireless.net/, 5 2010.

[104] S. Sengupta, S. Rayanchu, and S. Banerjee, "An Analysis of Wireless Network Coding for Unicast Sessions: The Case for Coding-Aware Routing," in *IEEE INFOCOM 2007*, 2007.

[105] S. Shakkottai, T. S. Rappaport, and P. C. Karlsson, "Cross-Layer Design for Wireless Networks," *IEEE Communications Magazine*, vol. 41(10), pp. 74–80, 2003.

[106] V. Srivastava and M. Motani, "Cross-layer Design: A Survey and the Road Ahead," *IEEE Communications Magazine*, vol. 43, no. 12, pp. 112–119, December 2005.

[107] R. Steinmetz and K. Nahrstedt, *Multimedia Fundamentals Volume 1: Media Coding and Content Processing*, Andrew Tescher, Ed. IMSC Press Multimedia Series, Prentice Hall PTR, 2002.

[108] R. Steinmetz and K. Nahrstedt, *Multimedia Systems*, Springer, Ed. Springer, 2004.

[109] R. Steinmetz and K. Wehrle et al., *Peer-to-Peer Systems and Applications*, R. Steinmetz and K. Wehrle, Eds. Springer, 2005.

[110] W. Stevens, "TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algoruthms," RFC 2001, 1 1997.

[111] W. R. Stevens, *TCP/IP Illustrated, Volume 1: The Protocols*, Addison-Wesley, Ed. Addison-Wesley, 1994.

[112] A. S. Tanenbaum, *Computer Networks*, Prentice Hall PTR, Ed. Prentice Hall PTR, 2002.

[113] L. Tassiulas and A. Ephremides, "Stability Properties of Constrained Queueing Systems and Scheduling Policies for Maximum Throughput in Multihop Radio Networks," *IEEE Transactions on Automatic Control*, vol. 37 (12), pp. 1936–1948, 1992.

[114] H.-Y. Wei, S. Ganguly, R. Izmailov, and Z. Haas, "Interference-aware IEEE 802.16 WiMax Mesh Networks," in *IEEE VTC 2005-Spring*, June 2005, pp. 3102–3106.

[115] M. Wieber, "Optimierte Verfahren der Bandbreiten-Verwaltung: IEEE-802.16 Mesh-Modus mit erweiterter Unterstützung für Network Coding," Master's thesis, TU Darmstadt, 2008.

[116] J. Wowra, "Evaluation of Network Coding Opportunities in Wireless Mesh Networks," 2009.

[117] G. Xylomenos and G. Polyzos, “TCP and UDP Performance over a Wireless Lan,” in *IEEE INFOCOM 1999*, 1999.

[118] G. Xylomenos, G. Polyzos, P. Mähönen, and M. Saaranen, “TCP Performance Issues over Wireless Links,” *IEEE Communications Magazine*, vol. 39, no. 4, pp. 52–58, 2001.

[119] Z. Ye, S. V. Krishnamurthy, and S. K. Tripathi, “A Framework for Reliable Routing in Mobile Ad Hoc Networks,” in *IEEE INFOCOM 2003*, 2003.

List of Figures

List of Tables

Appendices

A Degree of Scheduling Freedom Explained

As discussed briefly in Sec. 5.3.2 previous metrics designed to measure network coding gains in WMNs are not suitable for measuring the benefit from network coding in reservation based WMNs. In our prior work [80, 82] we have hence introduced a new means to model and measure the gains from network coding in a reservation based WMN using TDMA. The metric we use for the purpose builds up on the concept of the degree of freedom or scheduling degree of freedom which is defined next.

Definition A.0.1. ***Degree of freedom of a slot****: We define the degree of freedom or scheduling freedom of a slot as the number of types of activities, which may be scheduled in a particular slot given its current status. The degree of freedom of a slot is given by the function $\lambda(s)$ where s is the slot status. Where $\lambda(s)$ is defined as given by the Eq. (A.1).*

$$\lambda(s) = \begin{cases} 0 & \text{for s} \in \{uav\} \\ 1 & \text{for s} \in \{tav,\ rav\} \\ 2 & \text{for s} \in \{av\} \end{cases} \tag{A.1}$$

The values for $\lambda(s)$ reflect the scheduling possibilities a node has in a given slot. In slots with status *av* the node can either schedule a transmission or reception of data, i.e. it has two possibilities, hence $\lambda(av) = 2$. It follows that $\lambda(rav) = \lambda(tav) = 1$ (only one degree of freedom left at the node) and $\lambda(uav) = 0$ (node possesses no degree of freedom). From the above we can define the degree of freedom of the entire WMN for a given range of frames as the summation of $\lambda(s)$ for all s at all the nodes in the network. This total degree of freedom reflects the capability to set up additional transmissions in the WMN.

Given a WMN with a TDMA system similar to the IEEE MeSH mode the degree of freedom of the entire network can be considered to be the degree of freedom of all the slots at all the nodes in the WMN.

To get an intuitive idea of the above let us look at the simple example in Fig. A.1.

The shaded circles show the communication ranges of the two nodes n_1 and n_2 respectively. The nodes in the corresponding ranges form the set of neighbours of the nodes respectively. Nodes which belong to the neighbourhood of both the nodes n_1 and n_2 respectively (i.e. in set $Nbr(n_1) \cap Nbr(n_2)$) lie in the communication range of both the nodes. The metric degree of freedom is designed to be lightweight to compute and not require knowledge of the entire schedule neither complete knowledge of the network topology.

Now consider that the TDMA system used in our WMN has a total of M slots per frame and N nodes with no transmissions scheduled in the WMN. Assuming that there are no transmissions scheduled in the WMN then each slot in the system will be in a status *av*. Then we can compute the degree of freedom of the entire network by simply summing up the degree of freedom for the slots at each node. Considering just a single frame t we then have the total network degree of freedom of the empty network as given by Eq. (A.2).

$$TotFreeDof^t = \lambda(av) \times M \times N \tag{A.2}$$

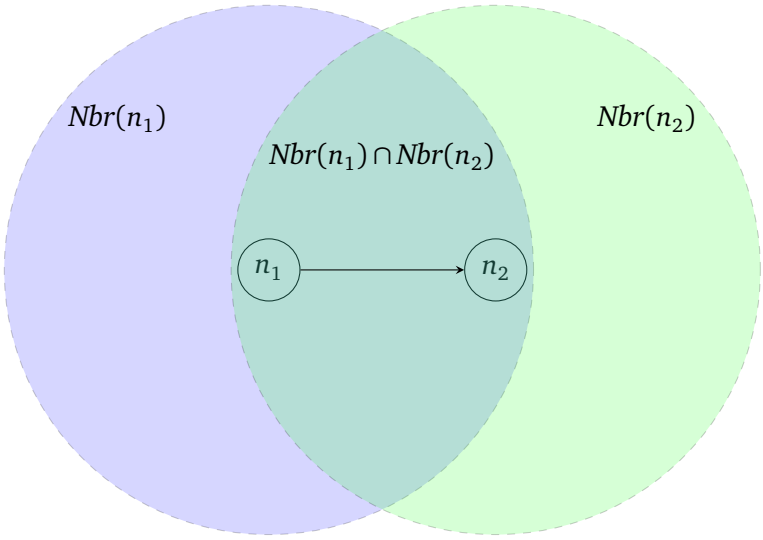


Figure A.1: Simple Topology to Illustrate Concept of Degree of Freedom

If the network is not free and there are some transmissions scheduled then similar to Eq. (A.2) we can compute the current degree of freedom for a given frame t as in Eq. (A.3).

$$CurrTotDof^t = \sum_{\forall n_i \in \mathcal{V}} \sum_{\forall s_i \in t} \lambda(status(s_i)) \tag{A.3}$$

Here, $status(s_i)$ gives the status of slot s_i, and G=($\mathcal{V}$, $\mathcal{E}$) is the graph representing the WMN. Now one can get a normalized value for the total network degree of freedom as in Eq. (A.4).

$$NormCurrTotDof^t = CurrTotDof^t / TotFreeDof^t \tag{A.4}$$

The higher the value of $NormCurrTotDof^t$ the more possibilities the network has for accommodating additional traffic in the network.

The above values Eq. (A.3) and Eq. (A.4) give a bird's eye view of the current status of the network as a whole. But at times it is also needed for a single node to find out the cost of scheduling a given transmission (e.g. for comparing the cost of a normal transmission versus transferring the same packets via network coding). Here, we can use the degree of freedom metric to allow a node to measure the cost of a transmission as the loss in the degree of freedom in the WMN due to a given transmission. Let $costDoF(n_x, \mathcal{E}', d)$ represent the cost for scheduling a transmission from node n_x on the set of edges $\mathcal{E}' \subseteq \mathcal{E}_{n_x}^{out}$ requiring d slots. Then, $costDoF(n_x, \mathcal{E}', d)$ can be computed as given by Eq. (A.7).

$$TxRcvSet = \{n_x\} \cup \{\cup_{e_i \in \mathcal{E}'} R_{e_i}\} \tag{A.5}$$

$$RcvSet = \{\cup_{e_i \in \mathcal{E}'} R_{e_i}\} \tag{A.6}$$

$$\begin{aligned} costDoF(n_x, \mathcal{E}', d) = {} & d \times (|Nbr(n_x) \backslash TxRcvSet| + |\{\cup_{n_i \in RcvSet} Nbr(n_i) \backslash TxRcvSet\}|) \\ & + 2d \times (|TxRcvSet|) \end{aligned} \tag{A.7}$$

As explained above we want to keep this computation lightweight such that it does not require complete knowledge of the network schedule nor the complete knowledge of the topology. Hence, we compute the cost of each transmission in the network considering that there is no other transmission scheduled in the network and all the slots in the network are in status *av* before the transmission is being scheduled.

Consider the simple example in Fig. A.1. Let us assume that we want to compute the transmission cost for scheduling transmissions on the link (n_1,n_2) shown by the arrow in Fig. A.1. Then from Eq. (A.7) we can see that the value for $costDoF(n_x,(n_1,n_2),d)$ is given by Eq. (A.8).

$$\begin{aligned} costDoF(n_x,(n_1,n_2),d) &= d \times (|Nbr(n_1)\backslash\{n_1,n_2\}| + |\{Nbr(n_2)\backslash\{n_1,n_2\}\}|) \\ &\quad + 2d \times (|\{n_1,n_2\}|) \\ &= d \times (|Nbr(n_1)\backslash\{n_1,n_2\}| + |\{Nbr(n_2)\backslash\{n_1,n_2\}\}|) \\ &\quad + 4d \end{aligned} \tag{A.8}$$

To understand the Eq. (A.8) better let us look at the Fig. A.1. For the d slots where the transmission on the link (n_1,n_2) is scheduled the nodes in the set $Nbr(n_1)$ are prevented from scheduling reception of data as these will be disturbed by the scheduled transmission, thus they lose one degree of freedom (i.e. to receive data) for each slot. Similarly the nodes in the set $Nbr(n_2)$ may not schedule any transmission of data in the scheduled minislots. Hence, these lose also one degree of freedom for each slot where the transmission is scheduled on the link (n_1,n_2). The nodes which lie in the intersection of the communication ranges of both the transmitter n_1 and the receiver n_2 have thus lost both the degrees of freedom for the scheduled slots. This is also the case for the nodes n_1 and n_2 as they lose the ability to schedule any further transmissions or receptions of data in these d slots. Similarly, using the Eq. (A.7) a node can compute the cost for the multicast transmissions too. It is easily seen that the node doing this needs only information about its two-hop neighbourhood, which it anyway has from the scheduling information at the MAC layer.

A node in the WMN which intends to serve as a relay node for a given network coding constellation can use the above metric to find out before activating a session for the network coding constellation whether network coding reduces the costs of transmission. Let us consider a network coding constellation $\mathfrak{N}^{n_r} = (n_r,\{\mathscr{S}_1,\mathscr{S}_2\})$. For this constellation the transmissions involved without network coding are: $Tx_1)$ $pred^{n_r}(\mathscr{S}_1) \rightarrow n_r$, $Tx_2)$ $n_r \rightarrow succ^{n_r}(\mathscr{S}_1)$ and the transmissions $Tx_3)$ $pred^{n_r}(\mathscr{S}_2) \rightarrow n_r$, and $Tx_4)$ $n_r \rightarrow succ^{n_r}(\mathscr{S}_2)$. As discussed above to compute the total cost of the transmissions we just sum up the cost of the four transmissions $costDoF(Tx_1) + costDoF(Tx_2) + costDoF(Tx_3) + costDoF(Tx_4)$*. Let *TotTxCost* be this sum. Let $pred^{n_r}(\mathscr{S}_1)$ be n_{p1} and $succ^{n_r}(\mathscr{S}_1)$ be n_{s1}, similarly let $pred^{n_r}(\mathscr{S}_2)$ be n_{p2} and $succ^{n_r}(\mathscr{S}_2)$ be n_{s2}.

Then *TotTxCost* is given by Eq. (A.13).

$$costDoF(Tx_1) = |Nbr(n_{p1})\backslash\{n_r\}| + |Nbr(n_r)\backslash\{n_{p1}\}| + 4 \tag{A.9}$$

$$costDoF(Tx_2) = |Nbr(n_r)\backslash\{n_{s1}\}| + |Nbr(n_{s1})\backslash\{n_r\}| + 4 \tag{A.10}$$

$$costDoF(Tx_3) = |Nbr(n_{p2})\backslash\{n_r\}| + |Nbr(n_r)\backslash\{n_{p2}\}| + 4 \tag{A.11}$$

$$costDoF(Tx_4) = |Nbr(n_r)\backslash\{n_{s2}\}| + |Nbr(n_{s2})\backslash\{n_r\}| + 4 \tag{A.12}$$

$$TotTxCost = costDoF(Tx_1) + costDoF(Tx_3) + costDoF(Tx_3) + costDoF(Tx_4) \tag{A.13}$$

* For simplifying the demand we assume unit demand (i.e. one minislot) for each of the above transmissions, it is trivial to obtain the equations for an arbitrary demand d.

When instead of the above four transmissions SONC is used by n_r for the constellation $\mathfrak{N}^{n_r}$ then the two transmissions Tx_2 and Tx_4 from the relay to the sink nodes n_{s1} and n_{s2} will be replaced by a single multicast transmission from the relay n_r to the sink nodes n_{s1} and n_{s2}. Let us denote this multicast coded transmission as Tx_{nc}.

The cost for the multicast coded transmission Tx_{nc} is given by Eq. (A.14).

$$costDoF(Tx_{nc}) = |Nbr(n_r)\backslash\{n_{s1}, n_{s2}\}| + |\{Nbr(n_{s1}) \cup Nbr(n_{s2})\}\backslash\{n_r, n_{s1}, n_{s2}\}| + 6 \tag{A.14}$$

We have two cases for the network coding constellation $\mathfrak{N}^{n_r}$, either the network coding constellation does not require overhearing or the network coding constellation does require overhearing for SONC to be deployed. We will consider the two cases separately:

- Case 1 No Overhearing Needed: Here, with network coding we will need to schedule the transmissions Tx_1, Tx_3 and Tx_{nc}. The total cost will network coding will then be the sum of the costs of these three transmissions. From Eq. (A.9), Eq. (A.11) and Eq. (A.14) and set theory we can easily see that the cost with network coding will always be lower than the cost without network coding as given by Eq. (A.13).
- Case 2 Overhearing is Needed: In this case we will schedule the network coding transmission Tx_{nc} as above, however, the either one of or both the transmissions Tx_1, and Tx_3 cannot be considered to be unicast transmissions anymore (depending on which of these transmissions need to be overheard). We can easily see from Eq. (A.7) that the cost of a multicast transmission is always equal to or greater than the cost of a unicast transmission (on one of the edges from the multicast edge set).
  Let Tx_1^m, and Tx_3^m denote the multicast (reduce to unicast transmission if no overhearing is needed for one of these) transmissions corresponding to the transmissions Tx_1, and Tx_3 which include corresponding edges to the nodes which need to overhear the original unicast transmissions. Now with overhearing the total cost with network coding will be the sum of the costs for the transmissions Tx_1^m, Tx_3^m, and Tx_{nc}. We however know from Eq. (A.7) that $costDoF(Tx_1^m) \geq costDoF(Tx_1)$ and $costDoF(Tx_3^m) \geq costDoF(Tx_3)$. We also know that $costDoF(Tx_{nc}) < (costDoF(Tx_2) + costDoF(Tx_4))$.
  With overhearing, the total cost of transmission is thus not guaranteed to be always less than the cost without using network coding. The decrease in the costs due to network coding may be outweighed by the increase in costs due to the additional multicast transmissions needed for overhearing. Hence, a relay node always computes the gains due to coding using the presented procedure in this case before deciding to activate network coding.

B Implementation Details

B.1 Algorithm for Computing $d_{\mathscr{S}_i}$

Given that we have a stream $\mathscr{S}_i$ at the relay node n_r then the value $d_{\mathscr{S}_i}$ is the minimum value such that for the considered window at time frame t for at least p_d percentage of the data rate measurements in the window the following equation: $|B_{n_r}^{\mathscr{S}_i}(t-k) - \mu_{\mathscr{S}_i}| \leq d_{\mathscr{S}_i}$, holds. Here $\mu_{\mathscr{S}_i}$ is the mean data arrival rate for the stream measured over the given window (see Sec. 5.3.2).

Algorithm 5 Computing $d_{\mathscr{S}_i}$ for stream $\mathscr{S}_i$

Definitions:
$B_{n_r}^{\mathscr{S}_i}$: Set $\{B_{n_r}^{\mathscr{S}_i}(t-(M-1)), B_{n_r}^{\mathscr{S}_i}(t-(M-2)), \ldots, B_{n_r}^{\mathscr{S}_i}(t)\}$ of data arrivals in bits at relay n_r for stream $\mathscr{S}_i$ for each frame in window [t-(M-1),t].
$\mu_{\mathscr{S}_i}$: Mean data arrival rate for stream $\mathscr{S}_i$ over the frames in the window [t-(M-1),t].
p_d: Percentage giving denoting the fraction of values from $B_{n_r}^{\mathscr{S}_i}$ which must satisfy $|B_{n_r}^{\mathscr{S}_i}(k) - \mu_{\mathscr{S}_i}| \leq d_{\mathscr{S}_i}$, $k \in [t-(M-1), t]$.
$d_{\mathscr{S}_i}$: The minimum value for this is to be computed.

procedure *algMinD*($B_{n_r}^{\mathscr{S}_i}$,$\mu_{\mathscr{S}_i}$,p_d)
  $B^{'} \leftarrow \emptyset$ ▷ $B^{'}$ is an array so multiple elements with the same value are permitted
  $j \leftarrow \emptyset$
  for $k = (t-(M-1))$ to $|t|$ **do**
    $B^{'} \leftarrow B^{'} \cup |B_{n_r}^{\mathscr{S}_i}(k) - \mu_{\mathscr{S}_i}|$
  end for
  $l \leftarrow 0$
  $d_{\mathscr{S}_i} \leftarrow 0$
  repeat
    $B_m \leftarrow min(B^{'})$ ▷ Gets the minimum value from the array $B^{'}$
    $d_{\mathscr{S}_i} \leftarrow B_m$
    $l \leftarrow (l+1)$
    $B^{'} \leftarrow B^{'} \backslash B_m$ ▷ Remove one element with value B_m from array $B^{'}$
  until $l/M \geq p_d/100$
  return $d_{\mathscr{S}_i}$
end procedure

The pseudocode for computation of $d_{\mathscr{S}_i}$ is given in Algo. (5).

B.2 Examples of Failure of the SONC Session Initialization Handshake

We have previously (in Sec. 5.3.3) looked at the handshake process for setting up a new network coding constellation in detail. In this section, we look at two representative instances of the handshake showing how the handshake can fail to achieve consensus about the slots to be reserved. In Fig. B.1 we show a successful handshake after an intermediate failure in achieving consensus. On the other hand Fig. B.2 shows an example where the handshake fails to activate a network coding session.

We first look at the example in Fig. B.1. Here the handshake runs normally up to the point where the proposal (specifying slots to be reserved for the multicast network coded transmission as well as slots for overhearing, where needed) is sent by the relay node n_r to the two sink nodes. Each sink node on receipt of the proposal will process it as discussed earlier in Sec. 5.3.3 with the aim to cross-check the availability of the slot ranges identified in the proposal. In the example in Fig. B.1 node n_{s_2} sends a reject message for the proposal. This is despite the fact that the slots in the proposal are a subset of the slots which it had earlier indicated as suitable in its response to the request from the relay node. The reason for this may be that in this time duration

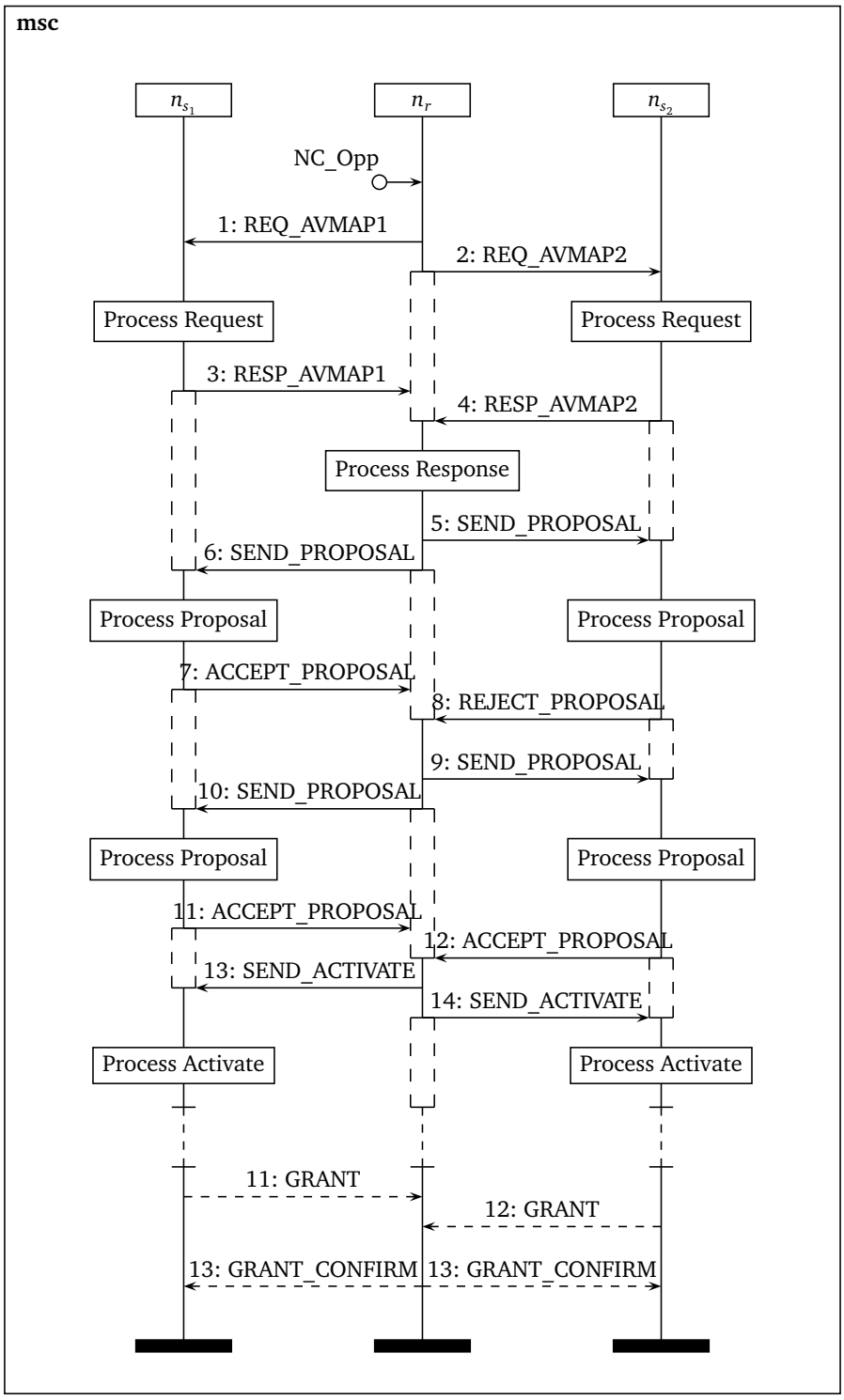


Figure B.1: Successful SONC Session Initialization Handshake After Intermediate Failure

(between sending of the response RESP_AVMAP2 and the sending of REJECT_PROPOSAL) the slots may have been blocked for transmissions in the neighbourhood of the node n_{s_2} such that the slot status for these slots (or a part of the slots in the proposal) does not permit the reception of the network coded transmission from the relay, or does not permit the overhearing of a sink node where needed.

The relay node n_r waits before getting an accept to its proposal from both the sink nodes before sending an activate message for the network coding session. However, in the example in Fig. B.1, it receives a reject from node n_{s_2}. If possible (as is the assumption for this example) the relay will then select another range of slots and issue a fresh proposal which it will send to both the sink nodes. In our example this new proposal is suitable for both the sink nodes and they both send accept messages to the relay node. The rest of the handshake then proceeds as normal to completion as discussed in Sec. 5.3.3.

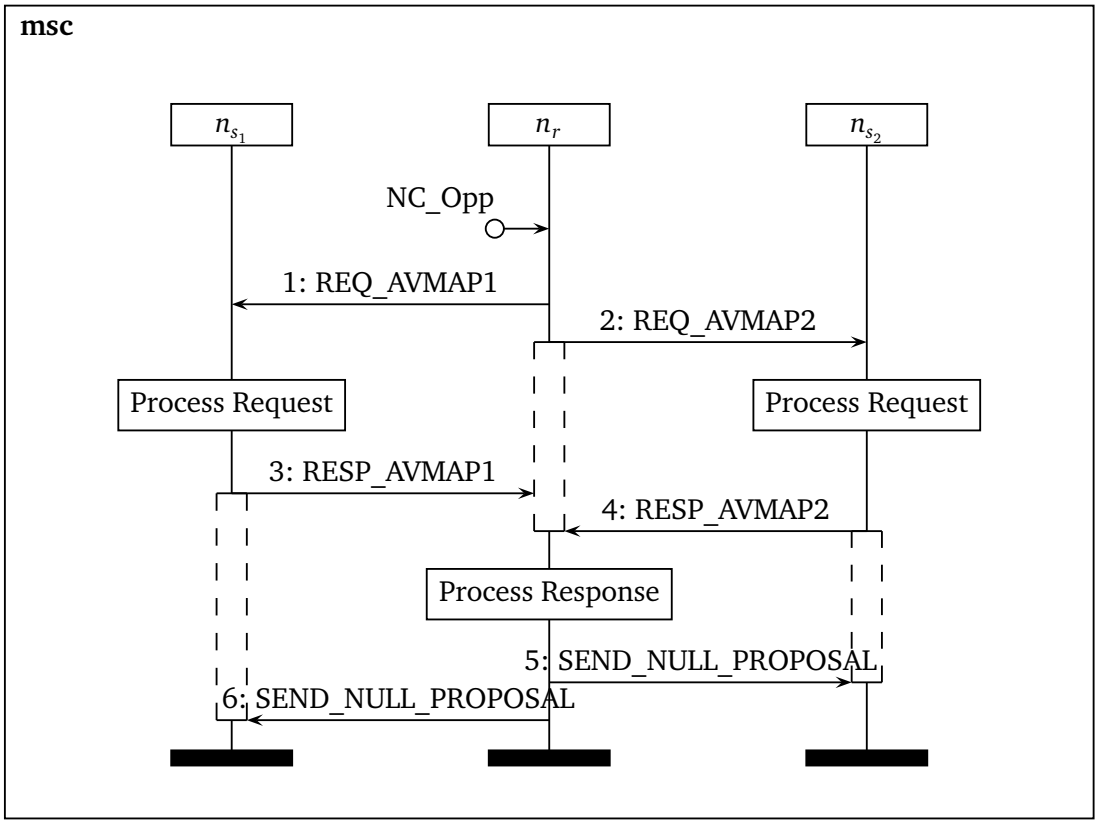


Figure B.2: Failure of SONC Session Initialization Handshake

The example in Fig. B.2 shows an example of the failure of the SONC session initialization handshake. In this example as in the normal case the sink nodes respond to the request messages from the relay by sending their individual response messages. These response messages are processed by the relay node n_r to identify slot ranges which are suitable to all the participants (n_r and the two sink nodes) for the network coded multicast transmission from the relay. It also tries to identify if slots for overhearing (where needed) could be successfully identified. If either no mutually suitable slots for the multicast network coded transmission are available or slots for overhearing (where needed) are not available then the network coding session cannot be activated at least for the current time period. Hence, the relay node indicates the end of the current handshake process with failure by sending a null (special indicator for no proposal) proposal to both the sink nodes. With this the handshake ends in the first phase without success and the nodes stop in the state NO_PROPOSAL_FOUND (see Sec. B.3). Both the state machines for this session will be deleted after waiting for a short number of frames.

B.3 State Machines for the SONC Session Initialization Handshake

To enable the sink nodes and the relay node to identify correctly how they should respond to the messages they receive during the SONC session initialization handshake we propose that these nodes maintain a state machine for each network coding session. Fig. B.3 shows the state machine for the relay node, and Fig. B.4 shows the state machine for the relay nodes. These are both finite state deterministic automata. The states are only relevant to the first phase of the SONC initialization handshake. The messages on the arrows indicate the signals which are sent to the state machine by the node in response to the messages exchanged during the handshake or in response to timer events. Thus, all the labels on the arrows are internal signals and not direct messages which are exchanged during the course of the handshake.

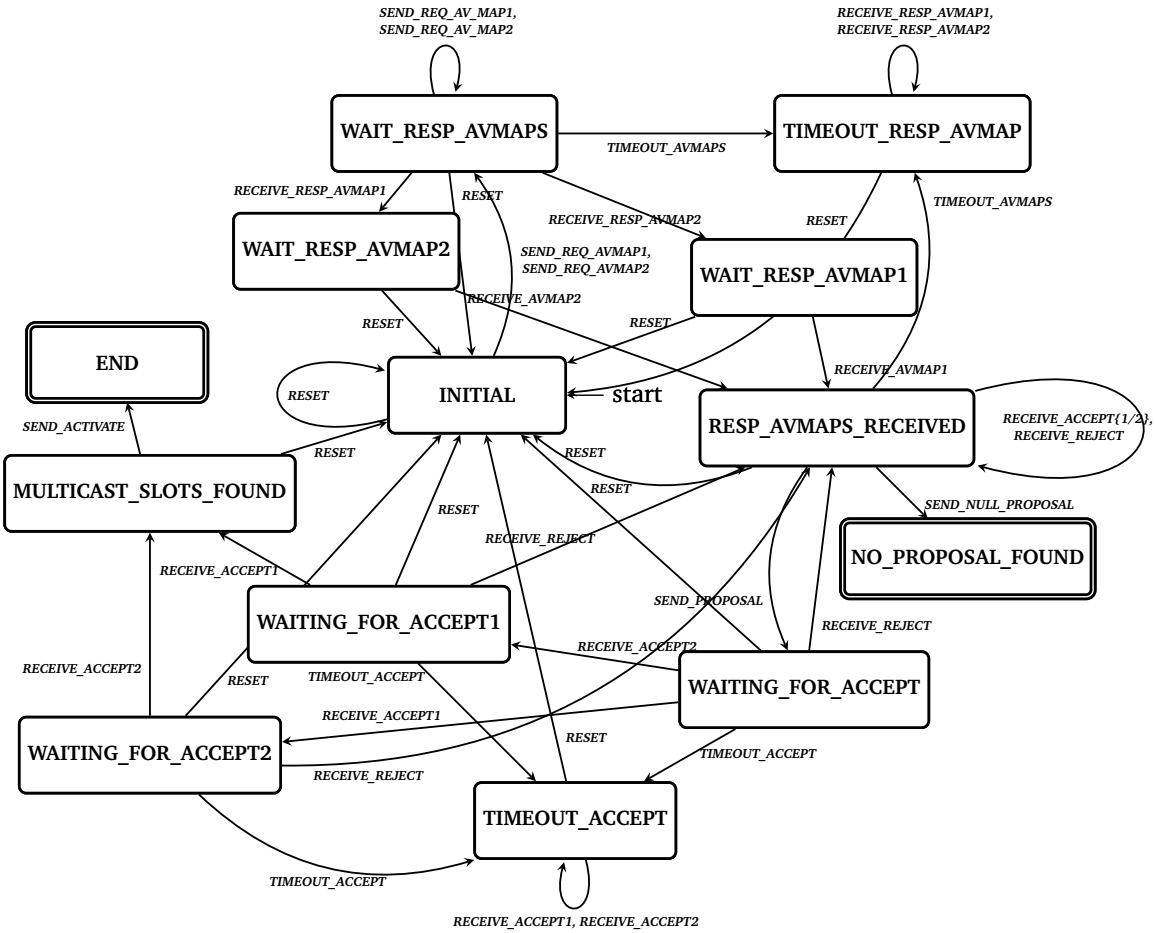


Figure B.3: State Machine for SONC Relay

Consider Fig. B.3, here the machine starts in the INITIAL state. After sending the request messages in the handshake (see Fig. 5.7) the relay will send the state machine the signals SEND_REQ_AVMAP1, and SEND_REQ_AVMAP2 and enter the WAIT_RESP_AVMAPS state. On correctly receiving response messages from both the sink nodes the state machine (via a number of signals) will reach the RESP_AVMAPS_RECEIVED state. After sending the proposal the state machine will be in the state WAITING_FOR_ACCEPT. Here, the state machine will wait for the individual accepts and on receiving an accept to the proposal the state machine will reach the state MULTICAST_SLOTS_FOUND (via a sequence of signals and states as shown in Fig. B.3). The negotiation of slots has now been successful and the relay will then issue a activate to both

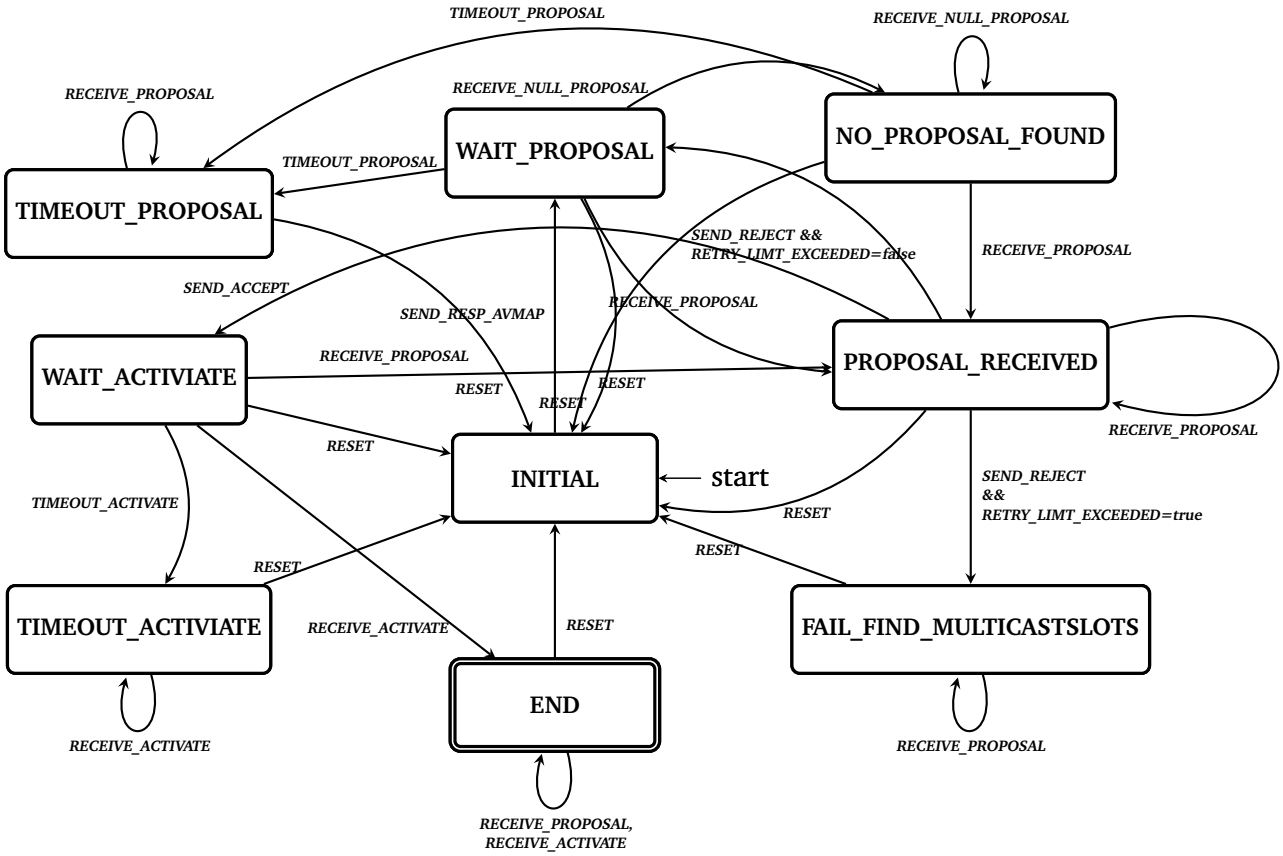


Figure B.4: State Machine for SONC Sink

the sink nodes and reach the END state of the state machine signalling the completion of the phase one of the SONC session initialization handshake. Similarly, in case of a failed handshake the state machine will end in the state NO_PROPOSAL_FOUND.

The state machine for the sink nodes works on similar principles and is shown in Fig. B.4 and is self explanatory. Hence we will not discuss this in further detail here.

B.4 Setup Details for Evaluation of SONC in Chap. 6

Basic experimental parameters for the base configuration of the IEEE 802.16 MeSH mode used in our simulation experiments in Chap. 6 can be found below. Table B.1 shows the PHY layer parameters employed in our simulations and Table B.2 shows the common simulation parameters.

B.5 CORE Message Formats

In this section we briefly outline the most important control messages used by CORE for its operation. For the following discussion we assume that the CORE server is located at the MBS, and use the terms MBS and CORE server interchangeably.

Table B.1: OFDM-Channelization-Parameter for the Air Interface According to ETSI-Specification and Channel Bandwidth of 14 MHz ([42], pp. 427, p. 810)

Variable	Value
Bandwidth (BW)	14,0 MHz
Subcarrier-Spacing (Δf)	62,5 kHz
Useful-Symbol-Time (T_b)	16 μs
Cyclic-Prefix/CP-Time (T_G)	1 μs
Symbol-Time (T_S)	17 μs
N_{FFT}	256
N_{used}	200
#of subchannels for data transmission	192

Table B.2: Common Configuration Parameters for Simulations in Chap. 6

Variable	Value
Simulated time	60 s
Frame length	20 ms
# of simulated frames	3000
OFDM symbols per frame	1176
OFDM sym. per TxOpp	7
OFDM sym. per Control Subframe	154
OFDM sym. per Data Subframe	1022
OFDM sym. per Minislot	4 (255x), 2 (1x)
# of TxOpps in Control Subframe	22
# of TxOpps for Dist. and Centr. Sched.	11
# of Minislots in Data Subfr. (max. 256)	256
Minislots for CS-Date (per Frame)	102
Minislots for DS-Data (per Frame)	154
Modulation	16-QAM, code rate = 1/2
# of bits per Minislotfor payload	1536 (i.e. 192 Bytes)

CORE-DEMUPDT

The CORE-DEMUPDT message (CORE Demand Update) is sent by the subscriber stations in the WMN to the MBS. We propose to send these messages as payload within User Datagram Protocol (UDP)[111, 112] messages. These messages are sent periodically, to inform the MBS of (significant) changes in the bandwidth demands of existing long-term flows, a significant change in demand also occurs in case a previously existing flow now ceases. The CORE-DEMUPDT message is also sent when a subscriber station detects a new flow, in both the cases the long term demand for a flow is reported to the CORE Server. See Sec. 8.3 for more details specifying when and how the demand updates for flows are sent to the CORE Server by the individual nodes in the WMN.

The aim of these messages is to keep the MBS informed at all times about the long-term flows in the WMN and also their respective bandwidth demand. To specify the flow we use the IP addresses of the source and destination of the flow. As only subscriber stations at which the flow enters the WMN send this message, to identify the source of the flow we use the source IP address in the header of the message carrying the CORE-DEMUPDT message to identify the source of the flow. Thus, there is only a field to specify the destination address. For identifying the destination of the flow we use the **DestAddress** field as shown in Tab. B.3. The **Bw. Demand** field identifies the long-term bandwidth demand of the flow in terms of minislots. A bandwidth demand of zero implies that the flow has ceased to exist. At present we are using the message format shown in Tab. B.3 for the study presented in this thesis.

Table B.3: Message Format for the CORE-DEMUPDT Message

Syntax	Size	Notes
Message Type = 0x01	8 bits	Identifier for message type
DestAddress	32 bits	Network address of flow's destination (IPv4 address)
Bw. Demand	8 bits	Bandwidth in minislots per frame

However, the message format may be extended in future to accommodate specification of further parameters.

CORE-RTUPDT

The CORE-RTUPDT (CORE Routing Update) message is sent by the MBS to subscriber stations whose routes need to updated based on the computations done by the CORE server.

Note that the CORE Server periodically receives updates about long-term flows in the WMN, and their bandwidth demands via the CORE-DEMUPDT messages it receives from the subscriber stations. The CORE server then runs its heuristics to find out and assign suitable routes for the flows, one route per flow. On computation of these routes the CORE server needs to notify the subscriber stations about the routing decisions they should make for packets belonging to these flows, additionally the subscriber stations also need to be notified of changes to the bandwidth reservations they should initiate corresponding to the changes in the flows and their routes. The CORE-RTUPDT message is used to inform the subscriber stations about new flows, updates to

existing flows or flows which have ceased to exist, and changes to their routes and bandwidth reservations.

Table B.4: Message Format for the CORE-RTUPDT Message

Syntax	Size	Notes
Message Type = 0x00	8 bits	Identifier for message type
Seq. Number	4 bits	One sequence number counter per flow $f(s,d)$. Seq. Number is increased by 1 for each new route update sent by the MBS to the nodes in the WMN corresponding to the flow $f(s,d)$
Bw. Demand	8 bits	Demand in minislots per frame
StartFrameNumber	12 bits	Frame number specifying the activation time (frame number) for the route update
AddressHop$_1$	32 bits	Network addresses (IPv4) of nodes on route. *AddressHop*$_1$ is the first hop (source) of the flow; *AddressHop*$_n$ is the last hop (destination) of the flow
AddressHop$_2$	32 bits	
AddressHop$_3$	32 bits	
⋮	⋮	
AddressHop$_n$	32 bits	

Tab. B.4 shows the format for the CORE-RTUPDT message. Similar to the CORE-DEMUPDT message we send these messages as a payload of UDP messages. As shown in Tab. B.4, each route update pertaining to a given flow is uniquely identified by a sequence number (**Seq. Number**). The sequence number field permits nodes to identify duplicate as well as outdated packets received and avoid needless updates to their routing tables. The CORE Server maintains a separate sequence number counter for each flow in the WMN, each time a route update message is generated for a particular flow and is sent to the nodes in the WMN, the corresponding counter is updated by 1.

Similar to the CORE-DEMUPDT message the **Message Type** field is used to identify the the type of CORE control message. The **Bw. Demand** field specifies in terms of minislots per frame the long-term bandwidth demand of the flow for which the CORE-RTUPDT message is being sent. The field **StartFrameNumber** is used to indicate to the nodes in the WMN the frame at which the new route being specified by the route update message is to be activated. This value will be stored by the nodes receiving and processing this message in the **Cross-Layer Database** and will be used for further processing as specified in Sec. 8.5.

We combine the specification with the specification of the new route for the flow. The route for a flow is specified as an ordered list of node addresses (either the IP addresses or the IEEE 802.16 node identifiers can be used) with the first address in the list ($AddressHop_1$) identifying the source of the flow and the last address in the list ($AddressHop_n$) identifying the destination node of the flow for which this route update is being sent. The interpretation of the list is such that packets belonging to the flow should travel along the nodes in the order specified by the route till they reach their destination. This implicitly specifies for each non-final node in the route a unique next-hop for the packets belonging to the flow.

Nodes receiving the route update message will check to see if their **CORE Routing Module** has a routing entry for the flow in question. If there is an entry, then it is updated based on the

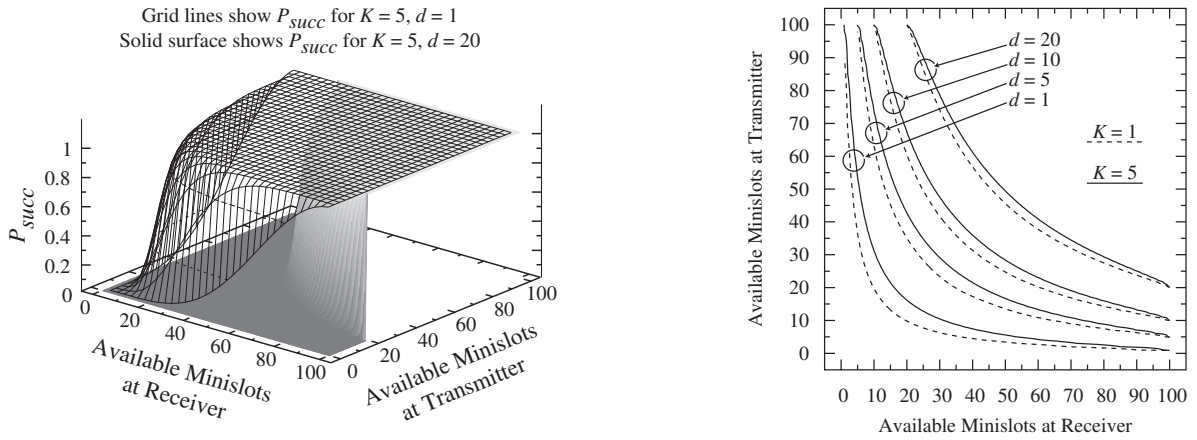
routing information contained in the message. Say we are at node n_i and node n_i is missing from the route (ordered list of addresses), however, there exists a routing entry for the flow in question at node n_i. This implies that node n_i will no longer be responsible for forwarding packets for this flow (according to the CORE extensions) after frame specified by the field **StartFrameNumber**. It will then note that it should delete the routing entry for the flow at that frame and reduce the demand for the flow on the previous outgoing link to zero.

On the other hand it can be the case that the node n_i has an existing entry for the flow in question in its **CORE Routing Module** and it also exists in the specified route. In this case it has to see if the specified next-hop (successor of n_i in the ordered list) is the same as the previous next-hop for the flow, and if the long-term demand being specified is the same as the previous demand registered for the flow. In case there are changes to any of the above the node then will take appropriate steps to change either/both route and bandwidth reservations as needed. The bandwidth reservation changes will be initiated in advance to the specified **StartFrameNumber**, whereas the route changes will be effective starting from frame **StartFrameNumber**. See Chap. 8 for details about this process.

The above format specifies that which we used for this work, and there are other means and formats possible for the message with extensions as needed.

C Additional Data and Results

C.1 Additional Results from the Analytical Model of the Three-way Handshake



(a) Effect of d on P_{succ}

(b) Contour Plots for $P_{succ} \geq 0.9$

Figure C.1: Plots Showing the Probability of Successfully Reserving d Slots in a Given Frame for Simultaneous Reception at K Neighbouring Nodes

Fig. C.1 shows the effect of an increase in the number of slots to be reserved on the probability of successful reservation. Fig. C.1(a) plots the variation keeping the number of neighbours to whom the bandwidth is to be reserved the same. Fig. C.1(b) shows the contour plots showing the lines beyond which (above right) the probability of successful bandwidth reservation is greater than ninety percent. The contour lines plotted for different demands and number of neighbours support the discussion in Sec. 4.1.1 showing how for increasing demand and number of neighbours to whom the bandwidth is to be reserved the number of slots which must be available at the transmitter and the receiver increases drastically to maintain the same probability of success.

C.2 Additional Results for Operation of CORE

In this section we will study CORE in operation for a small topology. The aim of the experimental setup is to study the performance of CORE under high network load. In particular we want to look at the rerouting done by CORE's heuristics in response to congestion in parts of the network irrespective of network coding. This highlights the fact that not only does CORE strive to increase the number of network coding opportunities in the network, but also routes the flows to reduce the overall interference (see Eq. (8.3) in the network.

The IEEE 802.16 standard parameters are the same as in Setup II (see Chap. 9, for which we have a total of 98 minislots in the data subframe. The frame duration is set to 10ms. At the nodes each outgoing link is allocated a buffer (for outgoing packets) of 64000 byte capacity. The network topology used is shown in Fig. C.2, the core server is at node A. The details for the

Flow	Src.→Dst.	Start	End	Demand
1	$G \rightarrow C$	9s	28s	20
2	$E \rightarrow F$	15s	24s	64
3	$A \rightarrow I$	21s	28s	51

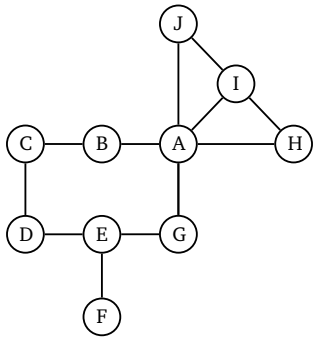


Figure C.2: Toy Topology and Flow Settings to Study CORE's Operation

flows are also shown in Fig. C.2 with the demand being specified in minislots per frame. *Start* and *End* columns specify the time the source nodes start and stop sending data, respectively. As in Setup II, CORE's central heuristics are allowed to use 70% of the total number of minislots in the system for computation of the schedules. The remaining slots are left aside to permit for short term reservations. We present results showing the buffer occupancy (also termed as "queue length" or "buffer size") for selected links in the topology. We also present results for the end-to-end delays for the flows. We use circled numbers to highlight important changes or values in the figures presenting the results. A particular circled number marks the same event in all the diagrams.

For our settings, we selected that the nodes will activate the new routes 40 frames after the corresponding packet was sent by the CORE server. Bandwidth is permanently reserved (1 minislot per frame) on the links (uplink/downlink) on the tree from the CORE server (also the mesh base station) to the nodes in the WMN. In practice bandwidth on these links will be usually available using centralized scheduling (if using the IEEE 802.16 MeSH mode). CORE control packets are forwarded with higher priority by the nodes compared to other data packets. Hence, these packets can be forwarded at least one hop per frame. The *OptRC* algorithm (see Sec. 8.4.2) is configured to give existing flows a higher priority than newer flows. This means that once a flow has been allocated bandwidth, it is not preempted by other flows. The flows are started after the network completes its initialization phase.

Consider the results in Fig. C.3 showing the end-to-end delays of the flows, and the results in Fig. C.4 showing the link status (queue length and Per_{∞} reservations) for selected important links in the network. The end-to-end delay for each of the three flows is initially (after the flows enter the network) very high (see (1),(3) and (5) in Figure C.3) with a corresponding increase in the queue lengths for the links (refer to Fig. C.4). This behaviour is due to the reservation based MAC of the IEEE 802.16 MeSH mode. Initially no reservations exist on the links on the path from the source to the destination of the flows. Till bandwidth is reserved packets must be buffered at each hop before they can be forwarded. In this phase CORE cannot be of much help as we want to use CORE only for stable long-term flows, and the bandwidth estimators at the nodes will take some time before they report the mean data rate of the flow to the CORE server (see the discussion in Chap. 8).

Flow 1 enters the network at time $t = 9$s and is initially routed using the path $G \rightarrow E \rightarrow D \rightarrow C$. This explains the increase in the queue length for the link (G, E) ((1) in Figure C.4(a)) and the correspondingly higher initial end-to-end delay (see (1) in Figure C.3).

The bandwidth reserved (minislots) for link (G, E) increases at approximately $t = 9.9$s, which corresponds to the first time the CORE server initiated bandwidth reservation for Flow 1 is activated. This leads to a decrease in the observed end-to-end delay for this flow (see (2) in

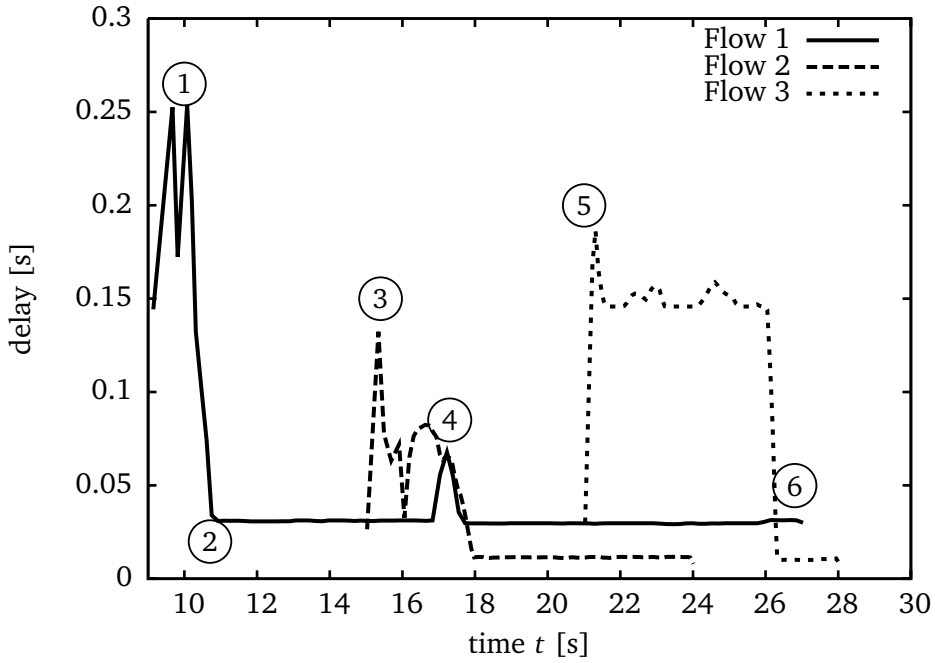


Figure C.3: End-to-end Delays with CORE

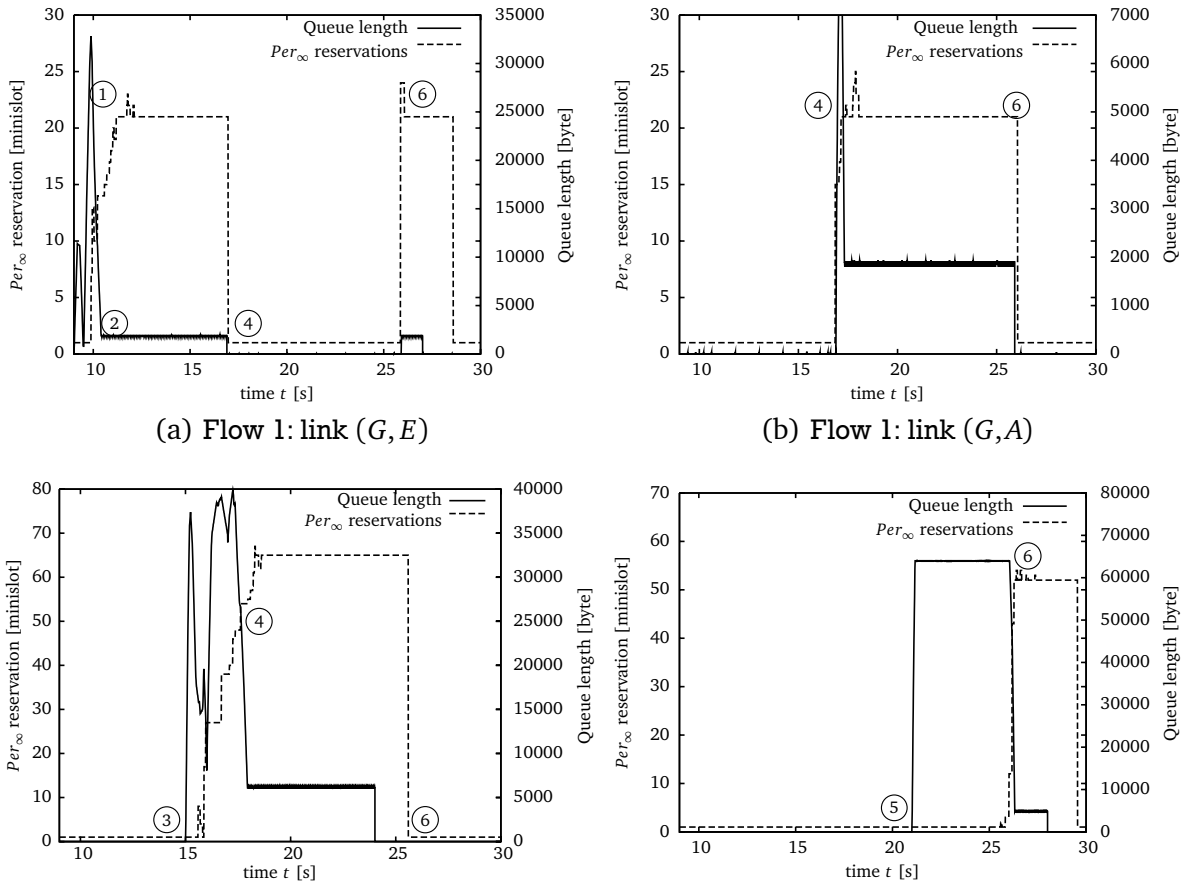
Figure C.3). The achieved delay of 30ms corresponds to the minimum achievable delay for a three hop route (assuming that a packet is not forwarded within the same frame it arrived at a relaying node).

Similar observations for the end-to-end delay and the link states can be observed for Flow 2 (see (3) and (4) in Figures C.3 and C.4(c)).

As the CORE server notes that node E cannot handle the demands of Flow 1 and Flow 2 simultaneously (the total demand would exceed the total of 70% of minislots available for planning the long-term reservations at the central server) it reroutes Flow 1 to the path $G \rightarrow A \rightarrow B \rightarrow C$. Therefore Per_{∞} reservations on link (G,E) are cancelled and an equivalent amount of minislots is reserved for link (G,A) (see (4) in Figures C.4(a) and C.4(b)) with persistence Per_{∞}. This reservation of bandwidth on the new path is initiated prior to activation of the new route (see Chap. 8), and hence the needed bandwidth is available on link (G,A) at the time of the reroute. Thus only a minor fluctuation in the delay of Flow 1 can be observed. (see (4) in Figure C.3).

Flow 3 enters the network at $t = 21$s. The end-to-end delay for Flow 3 does not drop down to an acceptable value after the expected initial peak (see ((5) in Figure C.3)) in the delay (unlike the case for Flow 1 and Flow 2). The reason being that the *OptRC* is set to give priority to existing flows over newer flows. As adding of Flow 3 to the existing Flow 1 would cause a congestion at node A, no Per_{∞} bandwidth reservation is initiated for this flow by the CORE server. Thus node A sends the packets of Flow 3 to I using short term reservations only. However, as sufficient minislots are not available, the buffer at link (A,I) is saturated soon after the start of Flow 3 (see (5) in Figure C.4(d)), which leads to the high delay of more than 150ms for just one hop. As additional packets are queued up and backlogged in the buffer for link (A,I) eventually packets have to be dropped. During the lifetime of Flow 3 we observed that nearly 10% of its packets are dropped.

The delay for Flow 1 is not affected despite the fact that it traverses the overloaded node A as bandwidth needed for this flow had already been reserved.



(a) Flow 1: link (G,E)

(b) Flow 1: link (G,A)

(c) Flow 2: link (E,F)

(d) Flow 3: link (A,I)

Figure C.4: Link States (Per_∞ Reservations, Queue Lengths) with CORE

Shortly after Flow 2 ceases at $t = 24s$ the CORE server reroutes Flow 1 back to its default route $G \rightarrow E \rightarrow D \rightarrow C$ a short time later. Hereby, needed bandwidth is reserved for the link (G,E) and cancelled for link (G,A) (see (6) in Figures C.4(a) and C.4(b)). This reroute helps Flow 3 whose delay drops at $t = 25.9s$ to a value of approximately 10ms (see (6) in Figure C.3) as additional network capacity is freed up for Flow 3. This allows the CORE server to initiate Per_∞ reservations for Flow 3 along with the above rerouting thereby leading to a lower end-to-end delay for Flow 3.

We observed no packet losses for Flow 1 and Flow 2, whereas nearly 10% of the data for Flow 3 had to be dropped due to buffer overflow.

Figure C.5 shows the end-to-end delays for the above scenario when CORE is not active. Here, each node in the network chooses the default routes for the individual flows. Where the default route for Flow 1 is $G \rightarrow E \rightarrow D \rightarrow C$. In comparison with the end-to-end delays with CORE (see Figure C.3) we observe that the delay of Flow 1 reaches a very high level of 200ms when the other flows are active. The end-to-end delay Flow 2 stabilizes at the value of approximately $120ms$. Only Flow 3 is able to achieve a relatively low delay (approximately 10ms), since it is the only flow that is not affected by other flows.

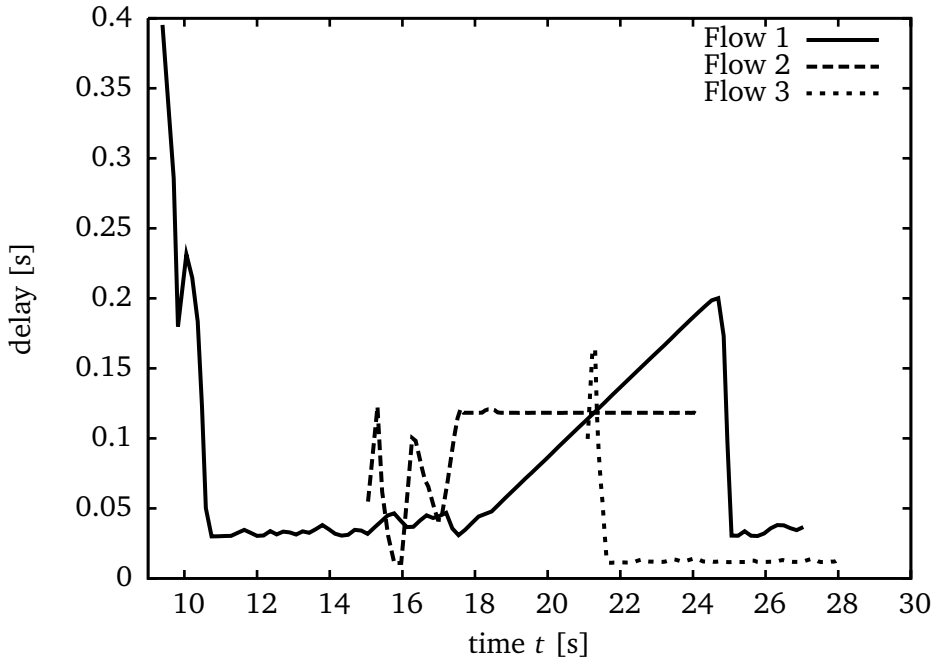


Figure C.5: End-to-end Delays without CORE in Operation

D Alphabetical List of Abbreviations

Nomenclature

av	Available (slot state)
rav	Receive Available (slot state)
tav	Transmit Available (slot state)
uav	Unavailable (slot state)
ACK	Acknowledgment
AODV	Ad hoc On Demand Distance Vector Routing
CBR	Constant Bit Rate
CORE	Centrally Optimized Routing Extensions
CP	Coded Packet
CPS	Common Part Sublayer
CS	Convergence Sublayer
CSCH	Centralized Scheduling
CSMA/CA	Carrier Sense Multiple Access/Collision Avoidance
DSCH	Distributed Scheduling
DSDV	Destination-Sequenced Distance-Vector Routing
DSR	Dynamic Source Routing
ETX	Expected Transmission Count
FTP	File Transfer Protocol
GSM	Global System for Mobile Communications
IE	Information Element
ILP	Integer Linear Programming
IP	Internet Protocol
LAN	Local Area Network
MAC	Medium Access Control
MAC PDU	MAC layer Protocol Data Unit
MaxSch	Maximal Scheduling Heuristic
MBS	Mesh Base Station

MDA	Mesh Deterministic Access
MeSH	Mesh mode of operation of the IEEE 802.16 standard
MILP	Mixed Integer Linear Programming
MSC	Message Sequence Chart
MSH-DSCH	MeSH Distributed Scheduling Message
NC	Network Coding
NCFG	Network Configuration Message
NP	Nondeterministic Polynomial
OptRC	Optimal Route Combination Heuristic
OSI	Open Systems Interconnect
P2P	Peer-to-Peer
PDU	Protocol Data Unit
PHY	Physical Layer
PINC	Pairwise Intersession Network Coding
PMP	Point to Multipoint
QoS	Quality of Service
RS	Relay Station
RSFH	Route Selection and Filtering Heuristic
RTS/CTS	Request To Send/Clear To Send
SAP	Service Access Point
SONC	Stream Oriented Network Coding
SS	Subscriber Station
SSA	Signal Stability-Based Adaptive Routing
STDMA	Spatial Time Division Multiple Access
TCP	Transmission Control Protocol
TDD	Time Division Duplex
TDMA	Time Division Multiple Access
UDP	User Datagram Protocol
UMTS	Universal Mobile Telecommunications System
VoIP	Voice over IP
WLAN	Wireless Local Area Network
WMN	Wireless Mesh Network
WNC	Wireless Network Coding

WRP	Wireless Routing Protocol
Xmt	Transmit

E List of Symbols

M	Total number of slots available for distributed scheduling per frame 34
M_{Tx}^{t}	Num. of slots suitable for scheduling transmission at the sender t (i.e. slots with status *av* or *tav*) 34
M_{Rcv}^{k}	Num. of slots suitable for reception at receiver k (i.e. slots with status *av* or *rav*) 34
P_{succ}	Probability of successfully reserving bandwidth for a network coding multicast in a given frame 35
P_{succ_f}	Probability of successfully reserving bandwidth for a network coding multicast in frame f 38
$f(s,d)$	Traffic flow originating at source node s, and with destination node d . . . 44
P_i^f	Denotes the i^{th} packet belonging to flow f 44
B_M	Denotes the number of bits which can be transmitted in a single minislot for the modulation in use in the WMN 45
$\mathscr{B}_t(P)$	Number of bits required (including headers) to transmit contents of packet P 45
$pred^{n_{relay}}(\mathscr{S})$	Denotes the previous hop node for packets belonging to stream $\mathscr{S}$ at the network coding relay node n_{relay} 46
$succ^{n_{relay}}(\mathscr{S})$	Denotes the next-hop node for packets belonging to stream $\mathscr{S}$ at the network coding relay node n_{relay} 46
$\mathscr{S}_{n_r}^{f_1,\ldots,f_n}$	Stream at node n_r composed of packets belonging to flows $f_1,\ldots,f_n$ where the previous and the next-hop of the packets are clear from the context and are identical for all packets 46
$\mathscr{S}_{n_r}^{n_{prev},n_{next}}$	Stream at node n_r composed of packets arriving from the previous hop n_{prev} and which are to be relayed on to node n_{next} 47
$P_i(\mathscr{S})$	Denotes the i^{th} packet belonging to stream $\mathscr{S}$ 47

$\hat{\in}$	Denotes the binary relation $\hat{\in} \subseteq \boldsymbol{P} \times \mathscr{C}$, where $p \hat{\in} c$ if the uncoded packet p is a component of the coded packet c, $\boldsymbol{P}$ is the set of packets and $\mathscr{C}$ is the set of coded packets 54
$pred^{n_{relay}}(P)$	Denotes the previous hop for packet P prior to its arrival at node n_{relay} . 55
$succ^{n_{relay}}(P)$	Denotes the next-hop for packet P to which it will be relayed by node n_{relay} 55
$S^{n_r}(p_x)$	Denotes the stream to which the packet p_x belongs to when it arrives at the relaying node n_r 55
$overhear(n_i)$	Set of nodes excluding the relay which need to overhear the uncoded transmissions from node n_i for a given network coding constellation at a given relay node 63
$sinkset(\mathfrak{N}^{n_r})$	Set of sink nodes for the network coding constellation $\mathfrak{N}^{n_r}$ 63
$sourceset(\mathfrak{N}^{n_r})$	Set of source nodes for the network coding constellation $\mathfrak{N}^{n_r}$ 63
$\mu_{\mathscr{S}_i}$	Denotes the mean data arrival rate for stream $\mathscr{S}_i$ at relay node n_r 66
$B_{n_r}^{\mathscr{S}_i}(t)$	Denotes the total bits arriving at the relay n_r for the stream $\mathscr{S}_i$ 66
G= $(\mathscr{V}, \mathscr{E})$	Graph G modelling WMN with set of vertices $\mathscr{V}$, and set of edges $\mathscr{E}$ 95
$\mathscr{V}$	Set of vertices $\mathscr{V}$ representing nodes in the WMN 95
$\mathscr{E}$	Set of edges $\mathscr{E}$ representing links in the WMN 95
(i, j)	Link (i, j) between nodes i and j in the WMN 95
T_e	Node which transmits on edge e 95
R_e	Node which is the receiver for transmissions on edge e 95
$\overline{e}$	Reverse edge corresponding to edge e 95
$\mathscr{E}_n^{out}$	The set of outgoing edges from node n 95
$\mathscr{E}_n^{in}$	The set of incoming edges to node n 95

$Nbr(n)$	Set of neighbours of node n 95
$\mathscr{P}_s^d$	Path, ordered list of one or more links which connect a pair of nodes s and d in the WMN 95
$S_{\mathscr{P}}$	Source node for path $\mathscr{P}$ 95
$D_{\mathscr{P}}$	Destination node for path $\mathscr{P}$ 95
$I(e)$	Set of edges which may not transmit (interfere) with transmissions on edge e 95
$\varsigma(e)$	Blocking cost of transmission on edge e 96
$\varsigma^{sd}(\mathscr{P}_s^d)$	Blocking cost of path $\mathscr{P}_s^d$ 97
$\mathscr{F}$	Set of flows indexed from 1 to $\lvert\mathscr{F}\rvert$ 97
$A(f_i)$	Indicator function signifying whether flow f_i can be allocated a feasible route or not 97
$V(f_i)$	Value or benefit associated by the network operator with admitting the flow f_i in the WMN 97
$\mathscr{R}_{f_i}$	Set of routes which are permitted based on the QoS specifications of the flow f_i 98
$r_{f_i}^j$	The j^{th} route $\in \mathscr{R}_{f_i}$ for flow f_i 98
$B_d^s(t)$	The total bits of data originating at node s with destination d in frame t 111
$Lt_d^s(t)$	Estimate for the long term bandwidth demand for flow f(s,d) in frame t 112
$\lambda(s)$	Denotes the degree of (scheduling) freedom for slot with status s 161
$status(s_i)$	Denotes the status of slot s_i 162
$costDoF(n_x, \mathscr{E}', d)$	Denotes the cost of scheduling a transmission for d slots from the node n_x on the set of edges $\mathscr{E}'$ 162

F Publications Listing

F.1 Publications as First Author

F.1:1. P. S. Mogre, N. d'Heureuse, M. Hollick, and R. Steinmetz. CORE: Centrally Optimized Routing Extensions for Efficient Bandwidth Management and Network Coding in the IEEE 802.16 MeSH Mode. *Wireless Communications and Mobile Computing, Special Issue: Architectures and Protocols for Wireless Mesh, Ad Hoc, and Sensor Networks*, Accepted for Publication.

F.1:2. P. S. Mogre, K. Graffi, M. Hollick, and R. Steinmetz. A Security Framework for Wireless Mesh Networks. *Wireless Communications and Mobile Computing, Special Issue: Architectures and Protocols for Wireless Mesh, Ad Hoc, and Sensor Networks*, Accepted for Publication.

F.1:3. P. S. Mogre, M. Hollick, J. Diaz Gandia, and R. Steinmetz. A Proportionally Fair Centralized Scheduler Supporting Spatial Minislot Reuse for IEEE 802.16 Mesh Networks. In *ICST QSHINE 2009*. 2009.

F.1:4. P. S. Mogre, M. Hollick, S. Dimitrov, and R. Steinmetz. Incorporating Spatial Reuse into Algorithms for Bandwidth Management and Scheduling in IEEE 802.16j Relay Networks. In *IEEE LCN 2009*. 2009.

F.1:5. P. S. Mogre, M. Hollick, C. Schwingenschloegl, A. Ziller, and R. Steinmetz. *WiMAX Evolution*, chapter WiMAX Mesh Architectures and Network Coding, pages 145–162. Wiley-Interscience, 2009.

F.1:6. P. S. Mogre, M. Hollick, R. Steinmetz, V. Dadia, and S. Sengupta. Distributed Bandwidth Reservation Strategies to Support Efficient Bandwidth Utilization and QoS on a Per-Link Basis in IEEE 802.16 Mesh Networks. In *IEEE LCN 2009*. 2009.

F.1:7. P. S. Mogre, N. d'Heureuse, M. Hollick, and R. Steinmetz. CORE: Centrally Optimized Routing Extensions for the IEEE 802.16 MeSH Mode. In *IEEE LCN 2008*. 2008.

F.1:8. P. S. Mogre, M. Hollick, M. Kropff, R. Steinmetz, and C. Schwingenschloegl. A Note on Practical Deployment Issues for Network Coding in the IEEE 802.16 MeSH Mode. In *IEEE WiNC 2008, IEEE SECON 2008*. 2008.

F.1:9. P. S. Mogre, N. d'Heureuse, M. Hollick, and R. Steinmetz. A Case for Joint Near-optimal Scheduling and Routing in TDMA-basedWireless Mesh Networks: A Cross-layer Approach with Network Coding Support. In *IEEE MASS 07*. October 2007.

F.1:10. P. S. Mogre, K. Graffi, M. Hollick, and R. Steinmetz. AntSec: Securing Organically Growing Wireless Mesh Networks. Technical report, Multimedia Communications Lab (KOM), TU-Darmstadt, Germany, March 2007.

F.1:11. P. S. Mogre, K. Graffi, M. Hollick, and R. Steinmetz. AntSec, WatchAnt and AntRep: Innovative Security Mechanisms for Wireless Mesh Networks. In *IEEE LCN 2007*. 2007.

F.1:12. P. S. Mogre, M. Hollick, N. d'Heureuse, H. W. Heckel, T. Krop, and R. Steinmetz. A Graph-based Simple Mobility Model. In *4th Workshop zu Mobilen Ad Hoc Netzen, KiVS 2007*. 2007.

F.1:13. P. S. Mogre, M. Hollick, C. Schwingenschloegl, and R. Steinmetz. *WiMAX/MobiFi: Advanced Research and Technology*, chapter QoS Architecture for Efficient Bandwidth

Management in the IEEE 802.16 MeSH Mode, pages 197–216. Auerbach Publications, 2007.

F.1:14. P. S. Mogre, M. Hollick, and R. Steinmetz. QoS in Wireless Mesh Networks: Challenges, Pitfalls, and Roadmap to its Realization. In *ACM NOSSDAV 2007*. 2007.

F.1:15. P. S. Mogre, M. Hollick, and R. Steinmetz. The IEEE 802.16-2004 MeSH Mode Explained, KOM-TR-2006-08. Technical report, KOM, TU Darmstadt, Germany, ftp.kom.tu-darmstadt.de/pub/TR/KOM-TR-2006-08.pdf, 2006.

F.1:16. P. S. Mogre. A Dynamic Network Architecture for Cellular Access Networks. In *KiVS 2005*. 2005.

F.2 Publications as Coauthor and Miscellaneous Publications

F.2:1. D. Christin, P. S. Mogre, and M. Hollick. Survey on Wireless Sensor Network Technologies for Industrial Automation: The Security and Quality of Service Perspectives. *Future Internet*, 2:96–125, 2010.

F.2:2. M. Graubner, P. S. Mogre, and T. Lorenzen. QoE Assessment for Audio Contribution Over IP (ACIP). In *AES 38th International Conference on Sound Quality Evaluation*. 2010.

F.2:3. M. Graubner, P. S. Mogre, and R. Steinmetz. A New QoE Model, Evaluation Method and Experiment Methodology for Broadcast Audio Contribution over IP. Technical Report KOM-TR-2010-2, TU Darmstadt, 2010.

F.2:4. M. Graubner, P .S. Mogre, R. Steinmetz, and T. Lorenzen. A New QoE Model and Evaluation Method for Broadcast Audio Contribution Over IP. In *ACM NOSSDAV 2010*. 2010.

F.2:5. A. Reinhardt, D. Christin, M. Hollick, J. Schmitt, P. S. Mogre, and R. Steinmetz. Trimming the Tree: Tailoring Adaptive Huffman Coding to Wireless Sensor Networks. In *EWSN 2010*. 2010.

F.2:6. A. Reinhardt, J. Schmitt, F. Zaid, P. S. Mogre, M. Kropff, and R. Steinmetz. Towards Seamless Binding of Context-aware Services to Ubiquitous Information Sources. In *CISIS 2010*. 2010.

F.2:7. F. Zaid, J. Schmitt, P. S. Mogre, A. Reinhardt, M. Kropff, and R. Steinmetz. Sorting the Wheat from the Chaff: Adaptive Sensor Selection for Context-aware Applications. In *IQ2S 2010, IEEE PERCOM 2010 Workshop*. 2010.

F.2:8. D. Christin, A. Reinhardt, P. S. Mogre, and R. Steinmetz. Wireless Sensor Networks and the Internet of Things: Selected Challenges. In *8th GI/ITG KuVS Fachgespräch on Wireless Sensor Networks*. 2009.

F.2:9. A. Ziller, P. S. Mogre, M. Hollick, and C. Schwingenschloegl. Reliable Broadcast Mechanism for the IEEE 802.16 Mesh Extension. In *4th IEEE BWA Workshop, IEEE Globecom 2008*. 2008.

F.2:10. K. Graffi, P. S. Mogre, M. Hollick, and R. Steinmetz. Detection of Colluding Misbehaving Nodes in Mobile Ad Hoc and Mesh Networks. In *IEEE Globecom 2007*. 2007.

F.2:11. M. Hollick, P. S. Mogre, C. Schott, and R. Steinmetz. Slow and Steady: Modelling and Performance Analysis of the Network Entry Process in IEEE 802.16. In *IEEE IWQoS 2007*. 2007.

F.2:12. M. Kropff, T. Krop, M. Hollick, P. S. Mogre, and R. Steinmetz. A Survey on Real World and Emulation Testbeds for Mobile Ad hoc Networks. In *IEEE Tridentcom 2006*. 2006.

F.2:13. C. Schwingenschloegl, P. S. Mogre, M. Hollick, V. Dastis, and R. Steinmetz. Performance Analysis of the Real-time Capabilities of Coordinated Centralized Scheduling in 802.16 MeSH Mode. In *IEEE VTC-Spring 2006*. 2006.

F.2:14. M. Hollick, P. S. Mogre, T. Krop, H. P. Huth, J. B. Schmitt, and R. Steinmetz. M^2DR: A Near-optimal Multiclass Minimum-delay Routing Algorithm for Smart Radio Access Networks. In *IEEE LCN 2005*. 2005.

F.2:15. T. Krop, M. Hollick, F. Krist, P. S. Mogre, and R. Steinmetz. Modeling Static and Dynamic Behavior of Routes in Mobile Ad hoc Networks. In *Poster, ACM MobiCom 2005*. 2005.

F.3 Patent Applications and Invention Reports

F.3:1. A Network Coding Mechanism for WiMAX Mesh Networks and Other TDMA-Based Mesh and Relay Systems. PriorArt Database.

F.3:2. Centrally Optimized Routing Extension. Patent Registration.

F.3:3. Detection of Colluding Misbehaving Nodes in Mobile Ad-Hoc and Wireless Mesh Networks. Patent Registration.

F.3:4. Efficient Availability Synchronization Scheme for IEEE 802.16 (WiMAX) Mesh and Relay Networks. PriorArt Database.

F.3:5. Efficient Bandwidth Utilization vis Indexed Bandwidth Reservation for Distributed TDMA Mesh Networks. Patent Registration.

F.3:6. Local Route Status Propogation for 802.16 Mesh Networks. Patent Registration.

F.3:7. Loop Avoidance for QoS-oriented Reactive Routing Schemes. Patent Registration.

F.3:8. Misbehaviour Detection in Wireless Mesh Networks without Promiscous Overhearing. Patent Registration.

F.3:9. Neural-Wavelet Filtering Optimized Bandwidth Reservation for Distributed Scheduling in TDMA Based Networks. Patent Registration.

F.3:10. Optimized MAC Layer Broadcast Mechanism for IEEE 802.16 based Mesh Networks. Patent Registration.

F.3:11. QoS Architecture for Efficient Bandwidth Management in the IEEE 802.16 Mesh Mode. Patent Registration.

F.3:12. Quality of Service Oriented Routing with Multipath. Patent Registration.

F.3:13. Reliable MAC Layer Broadcast Mechanism for IEEE 802.16 based Mesh Networks. Patent Registration.

F.3:14. Route Quality Estimation for 802.16 Mesh Networks with Cross-layer Metrics. Patent Registration.

F.4 Master/Diploma, Bachelor, and Student Theses Guided

[F.4: 1] Boris Bliwier, “Medium Access Control Layer Quality of Service Mechanisms for IEEE 802.16 Mesh Mode”, Diploma Thesis KOM-D-0259, 2006

[F.4: 2] Andreas Ziller, “Dienstgueteorientiertes Routing in IEEE 802.16 Mesh Netzen mit Cross-Layer Optimierung”, Diploma Thesis KOM-D-0232, 2006

[F.4: 3] Kalman Graffi, “A Security Framework for Organic Mesh Networks”, Diploma Thesis KOM-D-0249, 2006

[F.4: 4] Damir Sarac, “Security Mechanisms for IEEE 802.16 based Mesh Networks”, Diploma Thesis KOM-D-0248, 2006

[F.4: 5] Anton Previanto, “Design and Analysis of QoS Distributed Scheduling Algorithms for IEEE 802.16 Mesh Mode”, Master Thesis KOM-D-0258, 2006

[F.4: 6] Marcos Belchior, “Cross-layer Architecture to Enable QoS Aware Routing for Mesh Networks”, Master Thesis KOM-D-0286, 2006

[F.4: 7] Christian Schott, “Analysis of Security Mechanisms in IEEE 802.16”, Student Thesis KOM-S-0223, 2006

[F.4: 8] Matthias Kropff, “Network Coding for Bandwidth Management in IEEE 802.16 Mesh Networks”, Master Thesis KOM-D-0250, 2006

[F.4: 9] Martin Creutizger, “Optimization of Algorithms for Transmission of Multicast and Broadcast Data in IEEE 802.16 Based Mesh Networks”, Bachelor Thesis KOM-S-0234, 2007

[F.4: 10] Nico d’Heureuse, “Cross-Layer Bandwidth Optimization Scheme for IEEE 802.16 Wireless Mesh Networks”, Master Thesis KOM-D-0288, 2007

[F.4: 11] Guillaume Vernet, “Forecast-based Bandwidth Reservation Framework for IEEE 802.16 Mesh Networks”, Diploma Thesis KOM-D-0302, 2007

[F.4: 12] Jesus Diaz Gandia, “Analysis and Joint Optimization of Centralized and ”, Diploma Thesis KOM-D-0313, 2008

[F.4: 13] Eduardo Santana Leon, “Analysis and Evaluation of Multihop Communication in IEEE 802.16 WiMAX Networks”, Diploma Thesis KOM-D-0314, 2008

[F.4: 14] Viraj M. Dadia, “Bandwidth Reservation Strategies for Distributed Scheduling in the IEEE 802.16 Mesh Mode”, Master Thesis KOM-D-0315, 2008

[F.4: 15] Vesselina Dimitrova, “Joint Optimization of Centralized and Distributed Scheduling in 802.16 WiMAX Networks”, Master Thesis KOM-D-0338, 2009

[F.4: 16] Stefan Dimitrov, “Bandwidth Management in 802.16 Relay Networks: Algorithms for Admission Control, Scheduling with Spatial Bandwidth Reuse”, Master Thesis KOM-D-0337, 2009

[F.4: 17] Martin Wieber, “Optimierte Verfahren der Bandbreiten-Verwaltung: IEEE-802.16 Mesh-Modus mit erweiterter Unterstützung für Network Coding”, Diploma Thesis KOM-D-0339, 2008

[F.4: 18] Benjamin Otto, ‘Empirical Assessment of Distributed Connection-based Quality of Service Semantics for the IEEE 802.16 Mesh Mode‘”, Diploma Thesis KOM-D-0340, 2009

[F.4: 19] Johannes Wowra, “Evaluation Of Network Coding Opportunities in wireless Mesh Networks”, Bachelor Thesis KOM-S-0291, 2009

[F.4: 20] Ahmad Baig, “Implementation and Evaluation of Layer-2 Based Mobility and Forwarding Mechanism in Wireless Mesh Networks”, Master Thesis KOM-D-0342, 2008

[F.4: 21] Oliver Ebert, “Reserving Bandwidth for QoS Support in IEEE 802.16 Mesh Mode”, Diploma Thesis KOM-D-0349, 2009

[F.4: 22] Maxim Graubner, “Analysis of AoIP and Improvement of QoS for AoIP Applications in NGN Internet Architecture”, Diploma Thesis KOM-D-0360, 2009

[F.4: 23] Zheng Li, “Analysis of Bandwidth Management in Relay WiMax Networks in Comparison to PMP WiMax networks”, Student Thesis KOM-S-0309, 2009

[F.4: 24] Mostafa Bedair, “Performance Analysis of Mobile Radio Fast Retrial Aloha With Capture”, Student Thesis KOM-S-0350, 2010

[F.4: 25] Zheng Li, “iCompass: Algorithms for Improving the Accuracy of Electronic Compass in Indoor Settings”, Diploma Thesis KOM-D-0389, 2010

G Curriculum Vitae

Personal Data

Parag S. Mogre
Born 11^{th} September 1979 in Mumbai, India.

Address: Rundeturmstrasse 10
64283, Darmstadt, Germany.

Education Details

09/2004–present	To be obtained: Ph.D. in ETiT, TU Darmstadt, Germany.
07/2002–06/2004	Master in Computer Science, Indian Institute of Technology, Guwahati, India. Degree: Master of Technology (M.Tech.). Grade Average C.P.I. 9.94/10. Master's Thesis Topic: "Near-Optimal Multiclass Minimum-Delay Routing for Cellular Networks with Variable Topology", thesis completed at KOM, TU-Darmstadt, in scope of DAAD-IIT Sandwich Scholarship.
06/1997–06/2001	Bachelor of Engineering (Computer Science), University of Mumbai, India. Grade Average: First Class, 70.28% at B.E.

Academic Experience

08/2007–present	Doctoral candidate and research staff member at the Multimedia Communications Lab, TU Darmstadt, Germany.
07/2004–07/2007	Doctoral candidate at the the Multimedia Communications Lab, supported by a Doctoral Scholarship from the German Science Foundation (DFG), TU Darmstadt, Germany.
10/2004–present	Teaching Assistant for Masters Level course, Communication Networks II, at TU Darmstadt, Germany.
10/2005–present	Teaching Assistant for Masters Level course, Communication Networks III, at TU Darmstadt, Germany.
10/2004–present	Tutor for seminar with the topic "Future Trends in Internet Research", TU Darmstadt, Germany.
10/2004–03/2006	Tutor for "Praktikum and Project Seminar" (Programming Exercises and Seminar for scientific presentation), covering the topic "Design and Evaluation of Protocols for Mobile Communication", TU Darmstadt, Germany.
01/2003–05/2003	Teaching Assistant for Undergraduate Level Algorithms course in Computer Science and Engineering at Indian Institute of Technology, Guwahati, India.

Professional Experience

10/2009–present	Consultant for Sensor Technologies and Multimedia Technologies at KiMK Gmbh, Seeheim, Germany.
10/2004–07/2009	Principal Investigator for Research Collaboration with Siemens AG, CT IC2, München, Germany, in Topics Related to WiMAX, Wireless Mesh Networks and Industrial Sensor Networks using IP Technologies.
01/2002–06/2002	Software Developer, Prime Technologies, Mumbai, India.
08/2001–10/2001	Graduate Engineering Trainee in Software and Application Services Dept., GTL-Limited, formerly Global Tele-Systems Ltd., Mumbai, India.

Services to the Research Community

Reviewer	IEEE Transactions on Vehicular Technology, IEEE Transactions on Mobile Computing, ACM TOMCAPP, IEEE ISWCS, IEEE IWSOS, IEEE INFOCOM, IEEE FMN, KiVS, WMAN, Elsevier, PIMRC, IEEE VTC, WCNC.
Member of TPC	IEEE FMN, ACM MOBIHOC, KiVS WMAN, MMEDIA, IAMCOM.
Member of Editorial Board	International Journal on Advances in Telecommunications.

Scholarships and Awards

02/2009	Invention Achievement Award for the year 2007/2008 from the Department of Electrical Engineering and Information Technology, TU Darmstadt, Germany.
02/2005	Received the KuVS-ITG award for the best master's thesis for the year 2004 (candidates for the prize are recommended by research labs from universities in German speaking countries).
12/2004	Received the award for the best master's thesis for the year 2004 at the Multimedia Communications Lab (KOM), given by the KOM Förderverein.
07/2004–07/2007	Doctoral scholarship from the Deutsche Forschungsgemeinschaft (DFG) for pursuing Ph.D. studies at the Multimedia Communications Lab, TU Darmstadt, Germany.
2004	1^{st} rank at M.Tech. Computer Science and Engineering Dept., Indian Institute of Technology, Guwahati, with C.P.I. 9.94/10.
09/2003–05/2004	Scholarship from the DAAD (German Academic Exchange Academy) within the DAAD-IIT Masters Sandwich Programme for pursuing the master's thesis at the Multimedia Communications Lab, TU Darmstadt, Germany.
06/2002–08/2003	Scholarship from the Ministry of Human Resources and Development, India, for pursuing M.Tech. studies at the Indian Institute of Technology, Guwahati, India.

H Erklärung laut §9 der Promotionsordnung

Ich versichere hiermit, dass ich die vorliegende Dissertation allein und nur unter Verwendung der angegebenen Literatur verfasst habe. Die Arbeit hat bisher noch nicht zu Prüfungszwecken gedient.

Darmstadt, 2010

Parag S. Mogre, M. Tech.